SOFTWARE SUPPORT COST ESTIMATING MODELS: A COMPARATIVE STUDY OF MODEL CONTENT AND PARAMETER SENSITIVITY

THESIS

Kevin L. Brummert
First Lieutenant, USAF

Philip R. Mischler, Jr.
First Lieutenant, USAF

AFIT/GCA/LAS/98S-3

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCA/LAS/98S-3

SOFTWARE SUPPORT COST ESTIMATING MODELS: A COMPARATIVE STUDY OF MODEL CONTENT AND PARAMETER SENSITIVITY

THESIS

Kevin L. Brummert
First Lieutenant, USAF

Philip R. Mischler, Jr.
First Lieutenant, USAF

AFIT/GCA/LAS/98S-3

The views expressed in this thesis are those of the authors
and do not reflect the official policy or position of the
Department of Defense, the U.S. Government, or the model developers.

AFIT/GCA/LAS/98S-3

SOFTWARE SUPPORT COST ESTIMATING MODELS: A COMPARATIVE STUDY OF MODEL CONTENT AND PARAMETER SENSITIVITY

THESIS

Presented to the Faculty of the Graduate School of Logistics

and Acquisition Management of the Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cost Analysis

Kevin L. Brummert
First Lieutenant, USAF

Philip R. Mischler, Jr.
First Lieutenant, USAF

September 1998

Acknowledgments

This thesis would not have been possible without the support of some very key and influential people. We would like to take this opportunity to say, Thank You.

First and foremost, we would like to thank our families for their support and understanding during this thesis effort. Kevin thanks his wife, Holly, and daughter, Danielle. Philip thanks his wife, Ashley, his daughter, Ami, and his son, Philip. We understand and appreciate the sacrifices the families endured during this effort.

We also want to thank Professor Ferens, our advisor, Lt Col Giuliano, our reader, and Maj Hauck, our reader after Lt Col Giuliano's retirement for their support and guidance. Shirley Tinkler from SMC, Sherry Stukes from MCR, Bob Havens from SPR, Jim Otte from Price Systems, Karen McRitchie from Galorath Associates, Tony Collins from Resource Calculations Inc., and Cheri Cummings from the Naval Center for Cost Analysis were all very helpful in providing guidance for the various software estimating models. Without their support, this thesis effort would not have been possible.

We would also like to give a special thanks to Lt Col Giuliano for his support and belief in our capabilities to complete this thesis effort and the GCA program in general.

In closing, we would like to thank the entire 98S GCA class: Dave Bach, Rob Bickel, John "G" Glover, Brian Kehl, Brad McDonald, John Odum, Lance Whitfill, and Mike Wilson. Their support and stress relief options made our time at AFIT enjoyable.

Kevin Brummert
Philip Mischler

Table of Contents

Page

List of Figures

Figure Page

Figure Page

Figure Page

List of Tables

Table Page

Abstract

This research entailed a comparison of five software estimating models: PRICE-S, SEER-SEM, SoftCost-OO, SoftEst, and SPR KnowledgePLAN. The objective was to research software support cost differences between the software models. This research effort is a follow-up to the 1993 Coggins and Russell thesis, which concentrated on software development cost. The following major question areas were addressed: (1) How do the differences between the models impact the resulting cost estimates? (2) To what degree can we explain and adjust for the differences between cost models? (Coggins and Russell, 1993:ix).

The same hypothetical baseline test case used by Coggins and Russell in 1993 was used for this research effort. The baseline test case has three Computer Software Configuration Items (CSCI), where CSCI 1 is composed of 50 thousand (K) Source Lines of Code (SLOC) further subdivided into 2 Computer Software Components (CSC) composed of 20K and 30K SLOC; CSCI 2 is a single component of 80K SLOC; and CSCI 3 is a single component of 45K SLOC. All items were for flight avionics of a manned aircraft.

The differences between the models significantly impact the resulting estimates. Over the five models evaluated, a range of over $60 million occurred during a twenty year support period. The researchers can explain the differences in the models due to the different algorithms used, but were not able to normalize the models to achieve equivalent estimates. The researchers feel a typical user will not be able to normalize

separate models and should, therefore, concentrate on learning one or two models in detail. Different models are more appropriate depending on the task or project being estimated.

SOFTWARE SUPPORT COST ESTIMATING MODELS: A COMPARATIVE STUDY OF MODEL CONTENT AND PARAMETER SENSITIVITY

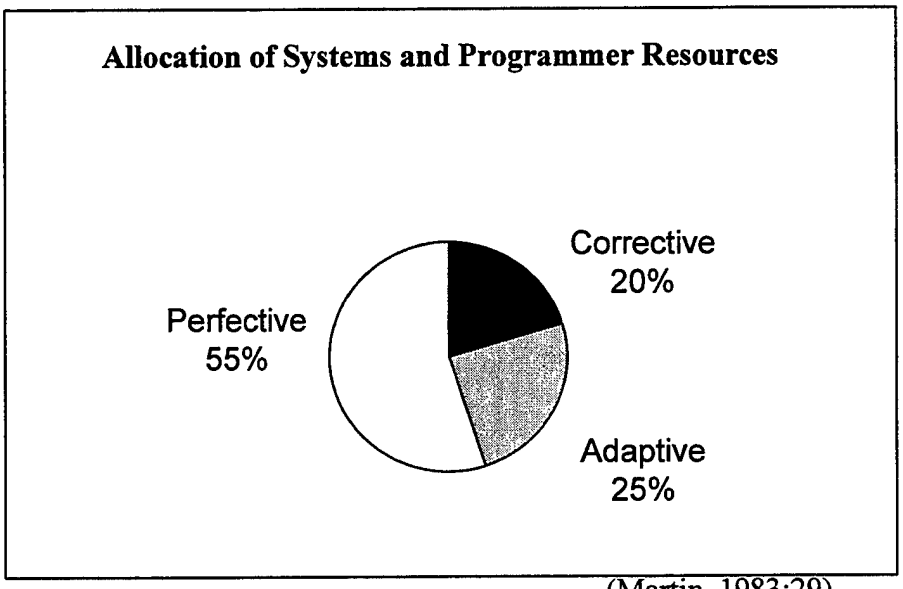
I. Introduction

General Issue

The computer has greatly enhanced our life and has given us the ability to transfer data/information across the world in literally a matter of seconds with a few keystrokes. PCs have become abundant in the surroundings that we live in and can be found in just about everyone's life. Whether it is in an individual's office, an elementary classroom, or the family recreation room, the PC has directly changed the way in which we lead our lives. In this same train of thought, the Department of Defense (DoD) has seen the use and role of the computer increase in carrying out missions supporting our national security interests and in preserving America's freedom. The DoD uses computer systems to control some of the most advanced weapon systems known to man. These include, the B-2 Stealth Bomber, the F-117A Stealth Aircraft, and the fighter of the future, the F-22 Raptor, to name a few. It has been estimated that the F-22 will have close to 7 million lines of code in the software system (Bischoff, 1991). This code will provide approximately 80% of the functionality of the aircraft (Fain, 1992). With so much of the functionality being built into the software of the F-22, it has become evident that the DoD is now depending on the usage of software in new weapon systems to provide the leverage that is needed to win wars, increase the warfighter's capability, allow more to be done with less, and to provide the flexibility in adjusting to unknown threats.

Software has become a crucial part and controlling aspect of most civilian and military systems. Civilian and military organizations have found that the best way to adapt existing systems to the ever changing requirements that they are faced with is to change the software that is running on the their systems. It's estimated that the DoD spends over $25 billion on software annually (Ferens, 1991:1). Over 60%, or $15 billion, of the expenses would be spent on software support alone (Ferens, 1991:1). The DoD is not the only organization with this type of expenditure support ratios. Hewlett Packard (HP) estimates 40-60% of production cost were directly related to maintenance alone (Coleman, 1994:44-49).

When the DoD looks at total Life Cycle Cost (LCC), over 70% of the LCC of software occurs in software logistics support (Ferens, 1992:4). Many times people hear the word "support" and simply think about "maintenance." "This is a misnomer, since software support involves much more than error correction" (Ferens, 1992:4). Software support normally contains three main areas: corrective support (performed to identify and correct software failures, performance failures, and implementation failures), adaptive support (performed to adapt software to changes in the data requirements or the processing environments), and perfective support (performed to enhance performance, improve cost effectiveness, improve processing efficiency, or improve maintainability). The interesting thing about the types of maintenance and associated percentages with a support effort is that the largest portion of the supportability being performed does not involve identifying and fixing problems with the software (Martin, 1983:29). Figure 1 illustrates the types of support and percentages.



(Martin, 1983:29)

Figure 1. Allocation of Systems and Programmer Resources

It is assumed within the three categories of maintenance costs that support (the activities used to explain system capabilities, to plan for future support, and to measure performance) can be found among the three main categories and is not a separate subcategory by itself (Martin, 1983:22). Ruetter further subdivided the three main categories of support into the following categories: corrective - corrective and emergency; adaptive - upgrades, changes in conditions and growth; and perfective - growth and enhancements (Martin, 1983:22). Table 1 illustrates Ruetter's categories. Note that growth of the software system was found in both adaptive and perfective maintenance activities and can be a major contributor to cost overruns.

Whether software is currently being developed or not, it is always in the process of being continually updated. Current trends show software development and support

Table 1. Examples of Maintenance Activities

Swanson's Categories	Ruetter's Categories
Corrective Maintenance	
	Correction to fix a switch
	Correction of failure to test for all possible conditions
	Correction of failure to process the last record in a file
Adaptive Maintenance	
	Implementation of a data-base management system (DBMS) for an Existing application system.
	Modification of designation codes from three characters to four Characters.
	Tuning a system to reduce response times.
	Converting the MRP system from batch to on-line operation.
	Adjustment of two programs to make them use the same record structures.
	Modification of a program to make it use a different terminal.
Enhancement Maintenance	
	Modification of the payroll program to incorporate a new union settlement.
	Addition of a new report in the sales analysis system.
	Improvement of a terminal dialogue to make it more user friendly.
	Addition of an extra column to a report.
	Adjustment of a program for printing bank statements to use a new design of preprinted stationery.
	Improvement of graphics output
	Adding an on4ine HELP command.
	Improvement of query processing capabilities to examine more types of data.
	Adding facilities required by auditors.

(Martin 1983:21)

cost for systems generally exceed the cost of the hardware system (Boehm, 1991:18). Past research has shown that changes in user requirements cause approximately 41% of the support costs, while hardware changes have only accounted for approximately 10% (Basset, 1995). A 1988 study shows software costs are predicted to grow at 12% per year (Boehm, 1988:1462). The majority of the increase in supportability costs has been

associated with a decrease in the productivity, with productivity decreases of 40:1 being reported (Boehm, 1981). These numbers lead one to believe that software support is the largest life cycle cost; and the ability to be successful in the future is going to be dependent on the manner in which software is supported "cost-effectively".

> The ability to continuously support our major software-intensive systems is a paramount mission requirement. Supportability is critical because there is always an inevitable need to correct latent defects, modify the system to incorporate new requirements, enhance the existing system to add capability, and alter it to increase performance. The ability to accommodate changes is an integral part of major software-intensive systems requirements. Unfortunately, when we have fielded insupportable systems, we have often had to expend the considerable time and funds necessary to provide the required support or we have had to abandon them altogether. We learned that it is far more cost-effective to address supportability as we define requirements, design the system, and plan for its operational life. (Department of the Air Force, 1996:11-1)

Thus, what is needed is to "reduce the risk of acquiring, managing, and maintaining software-intensive systems by ensuring that they can be modifiable, expandable, flexible, interoperable, and portable - i.e., supportable" (Department of the Air Force, 1996:11-1).

The United States is not the only country having problems with software. In Canada, "Little attention has been paid to software maintenance cost despite the fact that, over the life of the product, maintenance costs may be significantly higher than development costs. Despite the fact that maintenance is expensive, little or no attention is paid to the life cycle costing of software" (Vigder, 1994:40).

Specific Problem

No document or study has focused on or concentrated solely on the software support components of the various software cost models used by the DoD. Most documents or studies concentrate on the software development issues, without going into a lot of detail about the support features of those same models. The purpose of this

research is to provide one document that thoroughly explains the differences in the software support area for the five selected models. The differences would include: unique support inputs, development parameters which affect support, maintenance definitions, time span covered, and underlying assumptions in model equations/algorithms.

Research Objective

The objective of this thesis is to develop a consolidated document which highlights the differences in definitions, assumptions, and methodologies used by the PRICE-S, SEER-SEM, SoftCost-OO, SoftEst, and SPR KnowledgePLAN cost models and examines the impact of these differences on the resulting estimates as related to software support. To achieve this objective, the questions in Table 2 must be investigated.

Scope of Research

This research effort was undertaken to support the Air Force Cost Analysis Agency (AFCAA). Specifically, AFCAA requested a technical analysis of the PRICE-S, SEER-SEM, SoftCost-OO, SoftEst, and SPR KnowledgePLAN software support cost estimating models. As a result, only the given five models were chosen for this analysis, even though other models exist and are used by other various organizations within the DoD, they were not addressed within this research effort.

This effort did not research the estimating accuracy of the models. "Research was conducted with the intent of explaining the differences between the models and examining the impact of these differences on the resulting estimates. Cost analysts should consider the strengths and weaknesses of each model as well as availability of

information, time constraints, and the nature of the proposed project prior to selecting a specific model" (Coggins and Russell, 1993:4).

Table 2. Research Questions

(1)	What are the unique input parameters that directly affect the support costs?
(2)	What is the estimating methodology being used?
(3)	What are the underlying algorithms?
(4)	What is the underlying basis for the parameter value?
(5)	Is support broken out into support sub-categories?
(6)	What types of sub-categories are included for example: Preventive, Adaptive, and Corrective?
(7)	What time span is the maintenance/support option covering for the various models (5-20)?
(8)	Does a change in development method or language affect support costs?
(9)	What is the recommended estimating range?
(10)	What type of data/database was the current model calibrated to?
(11)	What differences exist between the cost models?
(12)	How do these differences impact the resulting cost estimates?
(13)	To what degree can we explain and adjust for the differences between the various cost models?

(Parallels Coggins and Russell, 1993:4)

Definitions

The following definitions are provided to insure a mutual understanding of key terms and concepts used in this thesis.

Adaptive. Alters existing software to address new requirements (enhancement) or new platforms (Tilley, 1996:7.2).

Algorithm. A mathematical set of ordered steps leading to the optimal solution of a problem in a finite number of operations (Stewart, 1987:557).

Analogy. An estimating methodology that uses actual cost of a similar past or present system and adjusts for complexity, technical, or physical differences to arrive at an estimate for the new system (Analytical Science Corporation, 1986:3.4.2).

Corrective. Removes faults or errors which escaped detection during development and testing of a system, or that were introduced during previous maintenance activities (Tilley, 1996:7.2).

Cost Estimating. "The art of collecting and scientifically studying costs and related information on current and past activities as a basis for projecting costs as an input to the decision process for a future activity." (Schwenke, 1992).

Cost Model. A tool consisting of one or more cost estimating relationships, estimating methodologies, or estimating techniques and used to predict the cost of a system or its components (Analytical Science Corporation, 1986:A-23).

CSCI, CSC, and CSU. Large software development efforts are generally broken down into smaller, more manageable entities called computer software configuration items (CSCIs). Each CSCI may be further broken down into computer software components (CSC) and each CSC may be further broken down into computer software units (CSU) (Department of Defense,1988: B-1 4).

Expert Opinion. An estimating methodology which queries technical experts and users regarding the estimated cost of a proposed system (Stewart, 1987:581).

Function Points. A software measurement technique presented in October 1979 by A.J. Albrecht, which considers the visible external aspects of software consisting of five items: inputs, outputs, inquiries by user, data files updated by application, and interfaces to other applications (Jones, 1995:2).

Hardware. Consists of the physical and electrical components of a computer system including items such as circuits, disk drives, wiring, and associated peripherals involved in the actual functions of the computer (SASET, 1990).

Normalization. The process of rendering constant or adjusting for known differences (Stewart, 1987:594).

Parametric Cost Model. A model that employs one or more cost estimating relationships for measurement of costs associated with the development of an item based on the project's technical, physical, or other characteristics (Stewart, 1987:596).

Perfective. Improving software attributes such as performance, memory usage, and documentation. Improves system without changing the basic system functionality (Tilley, 1996:7.2).

PRICE-S. Programmed Review of Information for Costing and Evaluation - Software. A commercial software cost-estimating model distributed by PRICE-Systems L.C.C..

Reengineering. Rebuilding a piece of software to suit some new purpose (to work on another platform, to switch to another language, to make it more maintainable, etc.); often preceded by reverse engineering. Examination and alteration of a subject system to reconstitute it in a new form. Any activity that improves one's understanding of software, or prepares or improves the software itself for increased maintainability, reusability, or evolvability (VanDoren, 1997).

Restructuring. Transformation of a program from one representation to another at the same relative abstraction level, usually to simplify or clarify it in some way (e.g., remove GOTOs, increase modularity), while preserving external behavior (VanDoren, 1997).

Reverse Engineering. The process of analyzing a system's code, documentation, and behavior to identify its current components and their dependencies to extract and create system abstractions and design information. The subject system is not altered; however, additional knowledge about the system is produced. Redocumenting and design recoveries are techniques associated with reverse engineering. Software complexity: some measure of the mental effort required to understand a piece of software (VanDoren, 1997).

SEER-SEM. System Evaluation and Estimation of Resources - Software Estimation Model. A commercial software cost estimating model developed by Galorath Associates, Incorporated.

SoftCost-OO. A commercial software cost estimating model developed by Resource Calculations Inc.

SoftEst. Software Estimator - Currently, a windows version of REVIC, a non-proprietary parametric cost estimating model based on Dr. Barry Boehm's Constructive Cost Model (COCOMO).

Software. The combination of computer programs, data, and documentation which enable computer equipment to perform computational or central functions (SASET, 1990).

Software Development Cycle. The software development cycle is typically broken into 8 phases: (1) System Requirements Analysis and Design, (2) Software Requirements Analysis, (3) Preliminary Design, (4) Detailed Design, (5) Code and CSU Testing, (6) CSC Integration and Testing, (7) CSCI Testing, and (8) System Testing. Software maintenance is often considered the ninth phase in this sequence (Department Of Defense, 1988).

Software Maintainability. Some measure of the ease and/or risk of making a change to a piece of software. The measured complexity of the software is often used in quantifying maintainability.

Software Support. Software does not break or wear out; support refers to corrective, adaptive, and perfective changes to software. Changes result when correcting software errors (corrective), responding to changing data or processing requirements (adaptive) and improving features through enhancements (perfective) (Ferens, 1992:4).

Source Line of Code (SLOC). For purposes of this research effort, SLOC is defined as all lines of executable and non-executable code with the exception of embedded comments (Coggins and Russell, 1993:7).

SPR KnowledgePLAN. A commercial software cost estimating model developed by Software Productivity Research.

Translation. Conversion of a program from one language to another, often as a companion action to restructuring the program (VanDoren, 1997).

Thesis Structure

The remainder of this research effort is directed at answering the investigative questions. The information gained by answering these questions will allow the researchers to compile a consolidated document, which highlights the differences between the support area of the software cost models and examines how these differences impact the cost estimates. Chapter II, Literature Review, reviews recent publications in the area of software support and describes each of the cost models selected for review. Chapter III, Methodology, explains how the research effort was structured to gather

information needed to answer the investigative questions. Chapter IV, Findings, analyzes the information obtained and answers the investigative questions. Chapter V, Conclusions and Recommendations, draws an overall conclusion regarding the differences between the software support areas of the cost models based on the literature review and information obtained and analyzed in the preceding sections of the thesis. Chapter V also identifies areas where further research may be warranted (Coggins and Russell, 1993:7-8).

II. Literature Review

Overview

This chapter examines recent publications, issues, and research in the field of software support. The chapter identifies issues which impact the accuracy of software support cost estimates, reviews estimating methodologies used for software support, explains a normalization technique for comparing different software cost models, and provides a summary of the estimating models used for this research effort.

Software Support Issues

"Software maintenance is a much maligned and misunderstood area of software engineering" (Pigoski, 1994). "There is seemingly insatiable demand for more functionality, interfaces that are easier to use, faster response, and fewer defects" (Stutzke, 1996:1). "In the software community, it is the software maintainer who gets the thankless job of fixing or enhancing someone else's program" (Arthur, 1997). "The Department of Defense and its contractors have arrived at a software crisis" (Brown, 1997) or as it has been most recently quoted as the "maintenance crisis" (Department of the Air Force, 96:11-5). These statements demonstrate the current situation of the DoD software community as of today. Discussed now are some of the important issues for software maintenance.

Software Support Risk. The cost to maintain a software product is from 60%-80% of total life cycle costs (Engle, 96). Software support is likely to be the largest life cycle cost driver and represents the major source of system risk that the DoD faces in major software acquisitions. Unforeseen requirements changes still carry high risk if they affect the old parts of the system. Cost and risk of maintenance of older systems are

further exacerbated by a shortage of suitable maintenance skills, analysts and programmers that are not trained to deal with these systems. "Industry wide, it is claimed that 75%-80% of all operational software was written without the discipline of structured programming" (Coleman, 1995). This would lead one to believe that there is a tremendous amount of risk in supporting the given software systems and in attempting to estimate the costs of the support effort.

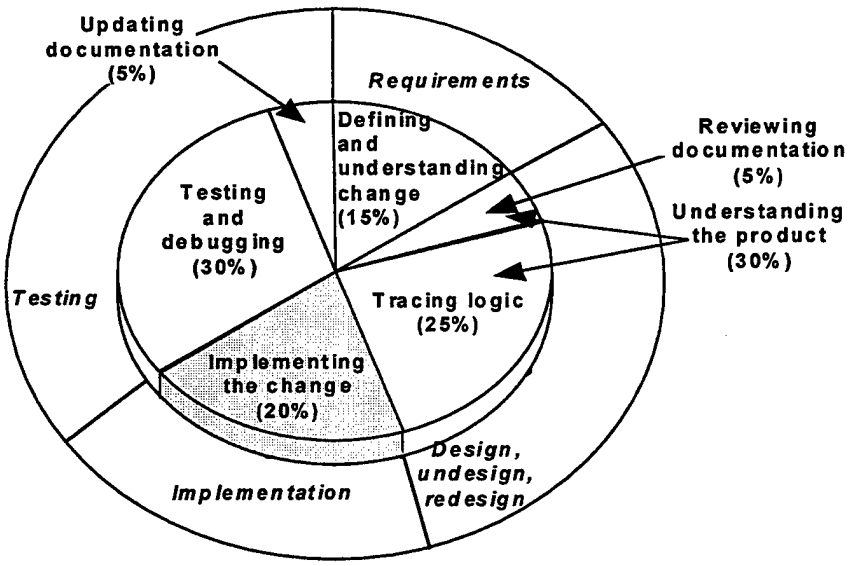
Impacts to Accuracy. "On most projects, development and maintenance phases are handled as two separate contracts. This creates an artificial division in the life cycle that makes it easy to overlook the costs of maintenance" (Engle, 1996: slide 37). Historically, the software life cycle has usually focused on development, but the majority of costs occur in the operational lifetime, so maintenance issues should be reflected in development practices (VanDoren, 1997:1). Movements are being made to try to improve the software support problem. DoD 5000.2-R recommends that "support concepts for new and modified systems shall maximize the use of contractor provided, long-term, total life-cycle logistics support" (Maibor, 1997:3.3.7).

> The major problem encountered by maintenance organizations when they try to arrive at accurate software estimates is the issue of requirements creep. Maintenance organizations are frequently calendar driven. Given a set of change requests and a delivery date, the organization estimates, which change requests, can be completed by the scheduled release date. However, during the course of the maintenance work there is a great temptation to add new change requests to the scheduled release without modifying the corresponding release date. This causes a great deal of frustration as the users do not get the promised changes, and developers are required to work to impossible schedules trying to meet the scheduled release as originally planned while including functionality not anticipated when the original schedule was created. The second problem with maintenance estimating is the large amount of overhead that is often associated with making a relatively small change. Typically, each change request is allocated to a designer to perform an impact analysis, which includes a cost estimate and a description of the impact on the design of

> the system. For minor changes, the effort to actually perform the estimate is most of the cost of designing and coding the change. If the designer has opened the source code and determined how the change is to be implemented in order to perform the estimate, most of the work of designing and coding is complete before it has been determined if or when the change is to be implemented. (Vidger, 1994:41)

Development factors affecting support. Software Support is different from software development in that a developer has a clean slate to work from in developing the software system. A person who performs software support, however, has to work with a product that has already been developed, has an existing infrastructure from which it was built from, and has given constraints built into it by the developer. Support and development are similar in the phases that are taken to arrive at the newly developed or updated product. "Support is the same as development because the maintainer must perform the same tasks as the developer, such as define and analyze user requirements, design a solution (within the constraints of the existing solution), convert that design into code, test the revised solution, and update documentation to reflect changes" (Department of the Air Force, 96:11-4). Figure 2 illustrates how support tasks correspond to and mirror the development process.

"So much of a system's cost incurred during its operational lifetime that maintenance issues have become more important and, arguably, this should be reflected in the development effort" (VanDoren, 1997). Therefore, the DoD needs to address the acquisition of software systems from a total life cycle perspective and not just solely on the development costs alone. Developing software that is easily supportable is one of the most important items in the equation for software success. "Process improvement programs have been shown in the report to reduce development costs and rework costs, as well as improve productivity, cycle time and quality" (McGibbon, 1996).

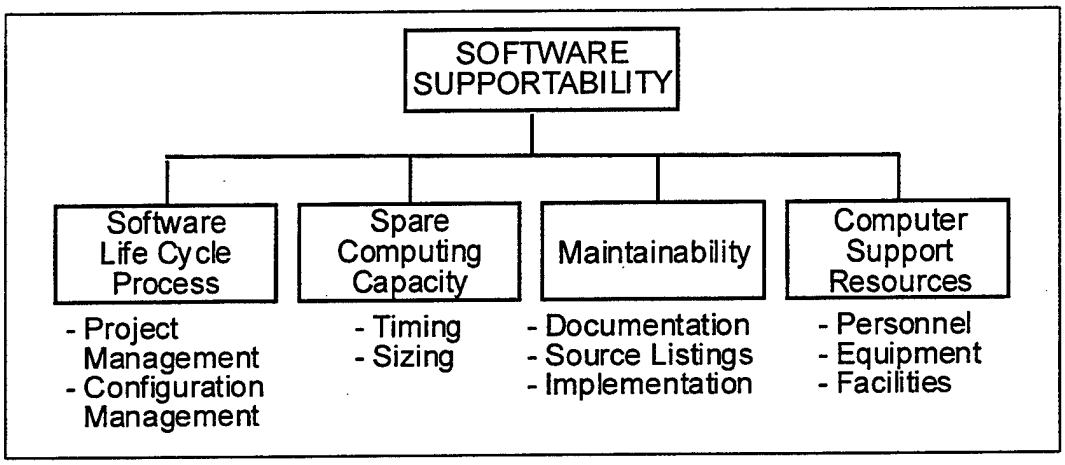


(Glass, 1992)

Figure 2. Support Tasks Superimposed on the Software Development Phase

A reduction of software support costs and the ability to streamline future investments will ultimately be the result of employing tools and techniques to effectively develop and maintain software systems. The past trend in software acquisition has been for the developer to build the program without advice or input from the individuals who will support the system. Once deployed, the software program is passed on to the support personnel who are left with the problem of how best to support the software. There are numerous factors within the development effort that affect the magnitude of the software support costs. These factors include software design, documentation, coding standards, and testing. The individuals or organization that is responsible for developing the software has to make a conscious decision to follow development practices that will allow the software system to be supported. It is when a development team is faced with schedule and cost constraints that the support factors within the development effort are often ignored. Another issue is the state of completeness that a system or program is at before it is ultimately turned over to the support personnel, who will then oversee the

operational capability of the system for the remainder of the system's lifetime. At times, an incomplete system may be turned over to the operational user due to the given cost and schedule constraints that the developer was faced with during the development effort. This leaves the first maintenance task to be a task of completing the unfinished system (Vidger, 1994:40). Figure 3 illustrates software supportability evaluation areas.

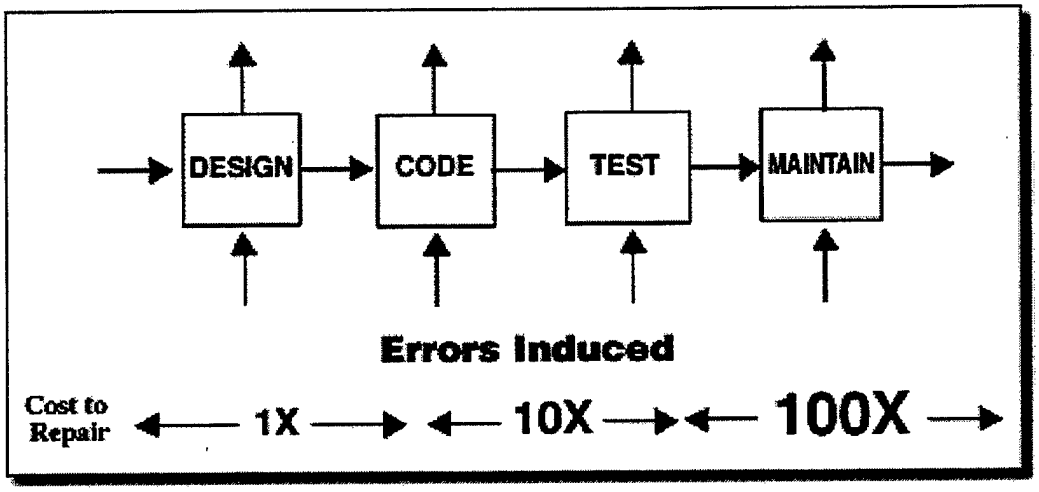


(Department of the Air Force, 1996)

Figure 3. AFOTEC Software Supportability Evaluation Areas

Some techniques are being employed today to reduce the amount of support cost required. A Maintainability Index (MI) is being tested (Welker, 1995). "Measurement and use of the MI is a process technology, facilitated by simple tools, that in implementation becomes part of the overall development or maintenance process. These efforts also indicate that MI measurement applied during software development can help reduce life cycle cost (VanDoren, 1997). Software Process Improvement (SPI) programs are being used which concentrate on "the detection and removal of software defects at or near the point of insertion of the defect" (McGibbon, 1996). Figure 4 indicates the benefits of finding a defect early in the development process. It is evident that, if the

defect is not caught until late in the process ("maintain" in Figure 4) the repair cost can be 100 times what would be required if the defect was found earlier in the design stages (McGibbon, 1996). This phenomenon is a key driver in effecting software support cost. Oklahoma City Air Logistics Center's Aircraft Software Division (LAS) has achieved a 90% defect reduction rate and a 26% average maintenance cost reduction using SPI, mainly driven by analytical procedures based on the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) (Belcher, 1996:1).

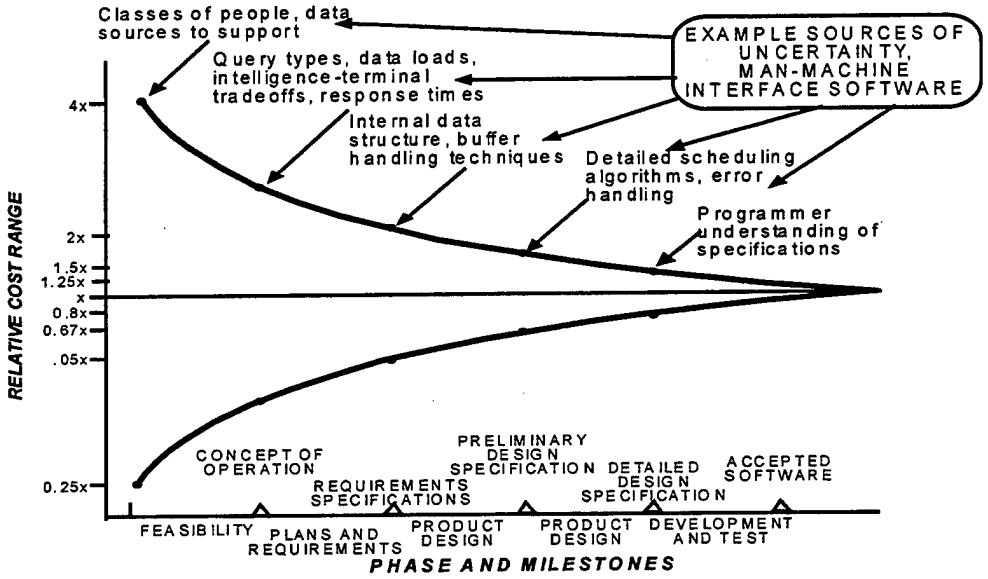


(McGibbon, 1996)

Figure 4. Modeling Rework Costs from Defects

At Tinker AFB, a return on investment of 6.35:1 was achieved from improvements recommended after their first Capability Maturity Model (CMM) appraisal. The key attributes associated with this success were the leadership portrayed by senior management, a recognition from everyone that process improvement is their job, and the ability to have visibility into progress. The researchers believe these same attributes would also affect the support cost. The CMM is seen as an excellent model of process changes that can be used to attain product improvement (McGibbon, 1996:9). All of the changes occurring in the software community place a cost estimator in a very

difficult position. An entire estimate, development through support, is required early in a program before all the goals and functionality of the software is known (Stutzke, 1996:1). Even when the top-level software design has been defined, the typical accuracy for the estimated schedule and effort has only been found to be within 25% of the final system costs (Boehm 81). Figure 5 shows how the accuracy of an estimate may be affected.



(Boehm, 1981)

Figure 5. Software Cost Estimation Accuracy Versus Phase

Estimating Methodology. The estimator has a choice of the type of an estimate, but must understand the estimate is only as good as the inputs used. Table 2 illustrates the estimating methodologies available for the estimator to use. Price to Win and Parkinson estimates are not considered options from the DoD perspective due to the integrity issues involved. Boehm also states Price to Win and Parkinson methods are not sound cost estimates and are unacceptable (Boehm, 1981:334).

Table 3. Cost Estimating Methodologies

Method	Description
Parametric/ Algorithmic Models	Provide one or more algorithms that produce a software cost estimate as a function of a number of variable which are considered to be major cost drivers
Expert Judgment	Involves consulting one or more experts, perhaps with the aid of an expert-consensus mechanism such as the Delphi technique
Analogy	Involves reasoning by analogy with one or more completed projects to relate their actual costs to an estimate of the costs of a similar new project
Parkinson	The principle that work expands to fill the available volume is invoked to equate the cost estimate to the available resources
Price to Win	The estimate is equated to the price believed necessary to win the job
Top-Down	An overall cost estimate for the project is derived from global properties of the software product. The total cost is then split among the various components
Bottom-Up	Each component of the software job is separately estimated, and the results are aggregated to produce an estimate for the overall job

(Boehm, 1981:329-330) (Marzo, 1997:9)

Boehm states that all estimating techniques have strengths and weaknesses and may complement one another, but none is better than the other from all aspects (Boehm, 1981:334). The bottom-up approach cannot be done "until there is a well-defined design and the nature and size of the components are known" (Wellman, 1992:31). The top down approach may overlook minor details and may also partition some cost to the wrong components. Expert judgment may be biased by the expert and can only be as accurate as the expert's knowledge allows. The analogy approach is only feasible when a similar program or project exists. Parametric/algorithmic models "are often unstable, in that small changes in certain sensitive input parameters can result in substantial changes in cost and schedule" (Ferens, 1996:29). Stutzke points out that, "No single estimating method is suited for every type of project" (Stutzke, 1996:20).

Software support costs are normally estimated by the parametric/algorithmic method. The main drivers pushing the use of parametric/algorithmic models are the time span for completing an estimate is normally fairly short, and the estimate is required very early in a program when detailed knowledge is not available (Ferens, 1996:29). Both of these factors favor parametric/algorithmic models, since the models allow analysts to generate quick estimates with limited inputs (Marzo, 1997:10). The parametric/ algorithmic method is not the only technique used in the DoD. When more detail is known about the system and support environment, the bottom-up approach becomes more commonly used.

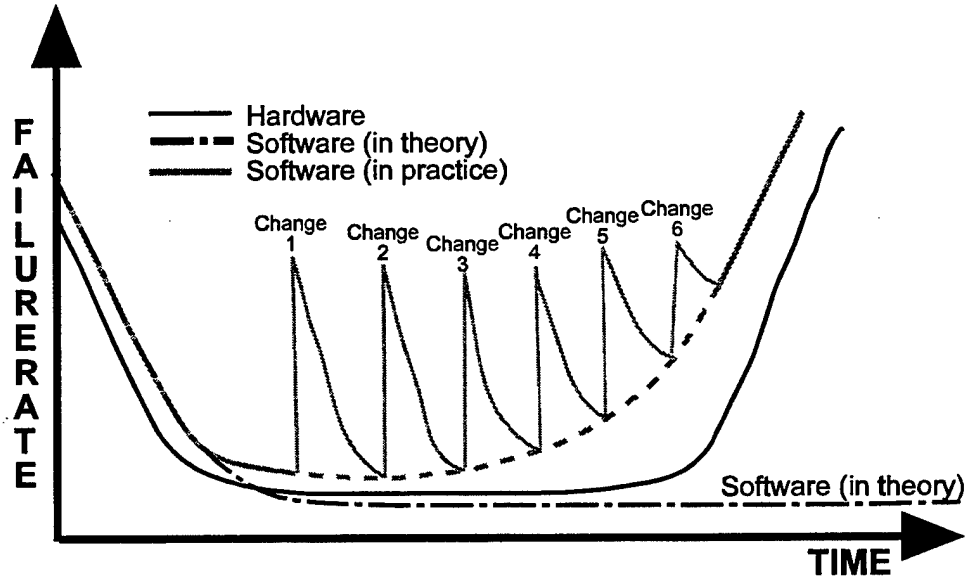
Software Support Characteristics

Software Support Phases. DoD systems have a long life span in comparison to commercial software systems. Over this long-term life span, the system will faces numerous changes from the point of initial deployment to the retirement of the system. As the software is supported and goes through the various support phases, the system begins to show some form of design entropy as the number of modifications to the system becomes increasingly large. As the system ages in time, the complexity of the system becomes greater and the ability to understand the conceptual underpinnings of the system becomes harder as time passes with the software being continually modified. The system often becomes fragile and brittle with the cost to accomplish changes increasing.

> While software does not wear out in the physical sense, it does deteriorate! There is an astounding difference when the software failure rate is superimposed on the bathtub curve. Like hardware, new software usually has a fairly high failure rate until the bugs are worked out. At which point, failures drop to a very low level. Theoretically, software should stay at that low level indefinitely because it has no tangible components upon which the forces of the physical environment can play. However, after software enters its operational life, it undergoes changes to

correct latent defects, to adapt to changing user requirements, or to improve performance. These changes make the software failure rate curve steadily begin an upward journey. Hardware deteriorates for lack of maintenance, whereas software deteriorates because of maintenance. (Glass, 1992)

By making changes, software maintainers often inadvertently introduce "side-effects" causing the defect rate to rise. Figure 6 illustrates the rising failure rate over time.



(Department of the Air Force, 1996)

Figure 6. Bathtub Curves for Hardware and Software

The following phasing information on pages 21-23 is extracted from VanDoren, *Maintenance of Operational Systems –An Overview*, 1997: 3-4.

Phase 1: The development or pre-delivery phase, when the system is not yet operational. Most of the effort in this phase goes into making Version One of the system function. But if total life cycle costs are to be minimized, planning and preparation for maintenance during the development phase are essential. Most currently operational systems did not receive this attention during development.

Requirements traceability to code. Requirements are the foundation of a system, and one of the most common faults of an operational system is that the relationship between its requirements and its code cannot be determined. Recovering this information for a system after it goes operational is a costly and time-consuming task.

Documentation and its usefulness in maintenance. The ostensible purpose of documentation is to aid in understanding what the system does, and (for the maintenance programmer) how the system does it. There is at least anecdotal evidence that

- Classical specification-type documentation is not a good primary source of information for the maintenance programmer looking for a problem's origin, especially since the documentation is frequently inconsistent with the code.

- The most useful maintenance information is derived directly and automatically from the code; examples include structure charts, program flow diagrams, and cross-reference lists. This suggests that tools that create and maintain these documentation forms should be used during development of the code, and delivered with it.

The complexity of the software. If the software is too complex to understand when it is first developed, it will only become more complex and brittle as it is changed. Measuring complexity during code development is useful for checking code condition, helps in quantifying testing costs, and aids in forecasting future maintenance costs.

The maintainability of the software. This is perhaps the key issue for the maintainer. The ability to measure a system's maintainability directly affects the ability to predict future costs and risks.

Phase 2: The early operational phase, when the delivered system is being maintained and changed to meet new needs and fix problems. Typically the tools and techniques used for maintenance are those that were used to develop the system. In this phase, the following issues are critical:

In a preventative maintenance regime, use of these types of measures will help establish guidelines about how much complexity and/or deterioration of maintainability is tolerable. If a critical module becomes too complex under the guidelines, it should be considered for rework before it becomes a problem. Early detection of problems, such as risk due to increasing complexity of a module, is far cheaper than waiting until a serious problem arises.

A formal release-based maintenance process that suits the environment must be established. This process should always be subject to inspection, and should be revised when it does not meet the need.

The gathering of cost data must be part of the maintenance process if life cycle costs are to be understood and controlled. The cost of each change (e.g., person-hours, computer-hours) should be known down to a suitable granularity such as phase within the release (e.g., design, code and unit test, integration testing). Without this detailed cost information, it is very hard to estimate future workload or the cost of a proposed change.

Phase 3: Mature operational phase, in which the system still meets the users' primary needs but is showing signs of age. At this point, the code has not been rewritten en masse or reverse engineered to recover design, but the risk and cost of evolution by modification of the system have increased significantly. The system has become brittle with age. It may be appropriate to assess the system's condition.

- The incidence of bugs caused by changes or "day-one errors" (problems that existed at initial code delivery) is rising, and the documentation, especially higher-level specification material, is not trustworthy. Most analyses of changes to the software must be done by investigating the code itself.

- Code "entropy" and complexity are increasing and, even by subjective measures, its maintainability is decreasing.

- New requirements increasingly uncover limitations that were designed into the system.

- Because of employee turnover, the programming staff may no longer be intimately familiar with the code, which increases both the cost of a change and the codes entropy.

- A change may have a ripple effect: Because the true nature of the code is not well known, coupling across modules has increased and made it more likely that a change in one area will affect another area. It may be appropriate to restructure or reengineer selected parts of the system to lessen this problem.

- Testing has become more time-consuming and/or risky because as code complexity increases, test path coverage also increases. It may be appropriate to consider more sophisticated test approaches.

- The platform is obsolete: The hardware is not supported by the manufacturer and parts are not readily available; the COTS software is not supported through new releases (or the new releases will not work with the application, and it is too risky to make the application changes needed to align with the COTS software).

Phase 4: Evolution/Replacement Phase, in which the system is approaching or has reached insupportability. The software is no longer maintainable. It has become so "entropic" or brittle that the cost and/or risk of significant change is too high, and/or the host hardware/ software environment is obsolete. Even if none of these is true, the cost of implementing a new requirement is not tolerable because it takes too long under the maintenance environment. It is time to consider reengineering

> When tasked with maintenance responsibility of legacy software which has become technologically obsolete, has deteriorated through years of changes, or must be changed anyway to work with new hardware or other software, it may be cost effective to re-engineer it. This involves systematic evaluation and alteration of an existing system to reconstitute it

(or its components) into a new form or converting it to Ada to perform within a new operational environment, to improve its performance, or to reduce maintenance costs. This process can combine several sub-processes, such as reverse engineering, restructuring, re-documentation, forward engineering, or re-targeting. (Department of the Air Force, 1996:11-1)

Preventative maintenance approaches. The approaches listed below are taken directly from VanDoren, *Maintenance of Operational Systems –An Overview*, 1997: 5-6. These approaches represent a few of the ways that current technology can help to enhance system maintainability.

Complexity analysis. Before attempting to reach a destination, it is essential to know where you are. For a software system, a good first step is measuring the complexity of the component modules

Functionality analysis. Function Point Analysis describes the uses and limitations of function point analysis (also known as functional size measurement) in measuring software. By measuring a program's functionality, one can arrive at some estimate of its value in a system, which is of use when making decisions about rewriting the program or reengineering the system. Measures of functionality can also guide decisions about where to put testing effort

Reverse engineering/ design recovery. Over time, a systems code diverges from the documentation - this is a well-known tendency of operational systems. Another phenomenon that is frequently underestimated or ignored is that (regardless of the divergence effect) the information required to make a given change is often found only in the code. Several approaches are possible here. Various tools offer the ability to construct program flow diagrams (PFDS) from code. More sophisticated techniques, often classified as program understanding, are emerging. These technologies are implemented as tools that act as agents for the human analyst to assist in gathering information about a programs function at higher levels of abstraction than a program flow diagram (e.g., retask a satellite).

Piecewise reengineering. If the system's known lifetime is sufficiently short, and if the evolutionary changes needed are sufficiently bounded, the system may benefit from a piecewise reengineering approach:

- Brittle, high-risk modules that are likely to need changes are identified and reengineered to make them more maintainable. Techniques such as wrappers, an emerging technology, are expected to aid here.

- For the sake of prudence, other risky modules are "locked," so that a prospective change to them can be made only after thoroughly assessing the risks involved.

- For database systems, it may be possible to retrofit a modern relational or object-oriented database to the system; Common Object Request Broker Architecture and

Graphic Tools for Legacy Database Migration describe technologies of possible use here.

Piecewise reengineering can generally be done at a lower cost than complete reengineering of the system. If it is the right choice, it delays the inevitable obsolescence. The downsides of piecewise reengineering include the following:

- Platform obsolescence is not reversed. Risks arising from the platforms software are unchanged- if the original database or operating system has risks, the application using them will also.
- Unforeseen requirements changes still carry high risk if they affect the old parts of the system.
- Performance may suffer because of the interface structures added to splice reengineered to old ones.

Transition / Restructuring / Modularizing. Translation and/or restructuring of code are often of interest when migrating software to a new platform. Frequently the new environment will not support the old language or dialect. Restructuring/modularizing, or rebuilding the code to reduce complexity, can be done simply to improve the code's maintainability, but code to be translated is often restructured first so that the result will be less complex and more easily understood. There are several commercial tools that do one or more of these operations, and energetic research to achieve more automated approaches is being done. Welker cites evidence that translation does little or nothing to enhance maintainability (Welker 95). Most often, it simply continues the existing problem in a different syntactical form; the mechanical forms output by translators decrease understandability, which is a key component of maintainability. None of these technologies is a cure-all, and none of them should be applied without first assessing the quality of the output and the amount of programmer resources required.

Test generation and optimization. Mission criticality of many DoD systems drives the maintenance activity to test very thoroughly. Boehm reported integration testing activities consuming only 16-34% of project totals (Boehm, 81), but recent composite post-release reviews of operational Cheyenne Mountain Complex system releases show that testing consumed 60-70% of the total release effort. Any technology that can improve testing efficiency will have high leverage on the system's life cycle costs. Technologies that can possibly help include: automatic test case generation; generation of test and analysis tools; redundant test case elimination; test data generation by chaining; techniques for software regression testing; and techniques for statistical test plan generation and coverage analysis.

Software Supportability Checklist. The following is a checklist to determine the supportability of a given software system.

Table 4. Software Supportability Checklist

1.	Maintainability	Requirement for a Maintenance Task Analysis (MTA)
2.	FTA, FMECA	Requirement for Fault Tree Analysis (FTA) and Failure Modes and Effects and Criticality Analysis (FMECA) to be performed to functional depth
3.	Defect Rate	Requirement to state a contractual target defect rate per lines of code over an agreed period including confidence limits
4.	Failure Identification	Design to provide features that achieve failure detection and location times
5.	Failure Snapshot	Design to provide features that achieve failure detection and location times
6.	Tool Kit	Provision of User/Maintainers software tool kits to aid failure location
7.	Loading and Saving Data	Design to allow loading or saving data in specified times
8.	Configuration Identification	User/maintainer able to identify the configuration status (version) without accompanying documentation
9.	Exception Handling	Design to allow exception handling to preclude failure conditions from aborting software during operations
10.	Support Policy Constraints	Use Study to include what the software must do and not do
11.	Support Maintenance Policy	Support specific maintenance policies and manpower ceilings and skill level availability to be stated
12.	Software Support and Maintenance Categories	Categories of software support and maintenance to be stated
13.	Media	Proposed media must: (a) suit the environmental requirements, and (b) be acceptable as a consumable item
14.	Media Copying	Simplify copying and distribution
15.	Media Marking	To allow physical and internal marking; safety critical items to be separately marked
16.	Packaging	Media packaging to be consumable, reusable, and robust
17.	Handling	Media to require no special precautions and meet Use study requirements
18.	Storage	Media to require no special precautions or facilities and meet Use Study requirements
19.	Transportation	Media and packaging to require no special requirements
20.	Training, User	User training required to detect failures and invoke exception handling
21.	Training Support	Support training required to detect and locate failures and invoke exception handling
22.	Publications	User and Support publications will be required
23.	Definitions	The Requirement must include contractually agreed upon definitions of: incident, fault, failure, defect, reliability, and failure categories
24.	Resources	Cost estimates must be sought for software maintenance
25.	Test Tools	Contractor-owned and maintained software test tools and documentation must be provided
26.	Test Tool Access	Access to test tools to be provided to software support personnel
27.	Incident/Failure Reporting	Incident and failure reporting to be available

(Department of the Air Force, 1996)

Normalization Explained

Normalization is simply the process of understanding and making adjustments for known differences between the models, whether estimated cost or schedule figures. Not all models estimate the same phases of the software development or support environment. Some models give a top level support cost or schedule, while others give a detailed breakout of the three major phases of software support: corrective, adaptive, and perfective. Understanding the mix (%) of these areas is extremely important to fully understand what information the model is actually providing.

Proper use of any cost model requires a thorough analysis of the model's assumptions and limitations, as well as its capabilities and features (The Analytic Sciences Corporation (TASC), 1986).

The *AFSC Cost Estimating Handbook* points out several key questions the analyst should resolve before using any model to prepare an estimate. Specifically:

(1) Is the data required to use the model available?

(2) What units of measure are used by the model (dollars or person-months)?

(3) What is the content of the database on which the model was derived?

(4) What is the range of input values for which the model is valid? (Analytic Sciences Corporation, 1986:8-6 - 8-10).

Unless the analyst fully understands the assumptions and phases included in each model, it is impossible to compare one model with another model (Coggins and Russell, 1993:15).

Cost Model Descriptions

The material describing the SEER-SEM, PRICE-S, SoftCost-OO, and Softest software cost estimating models on the following pages (28 – 37) was extracted directly

from the *PARAMETRIC COST ESTIMATING HANDBOOK -- Joint Government /Industry Initiative,* Department of Defense, Fall, 1995 with only minor modifications.

SEER-SEM

SEER-SEM is part of a family of software and hardware cost, schedule and risk estimation tools. SEER models run on IBM, Macintosh, and Sun/UNIX platforms with no special hardware requirements. SEER-SEM is used throughout the aerospace and defense industry on two continents. All issues found in today's software environments are addressed.

Inputs

SEER-SEM accepts source lines of code (SLOC) or function points or both. When selecting function points, the user may use IFPUG standard function points or SEER function-based inputs, which include internal functions. Users follow a Work Breakdown Structure (WBS) describing each CSCI, CSC, and CSU (module or element) to be estimated. Knowledge bases are used to provide fast and consistent inputs describing complexity, personnel capabilities and experience, development support environment, product development requirements, product reusability requirements, development environment complexity, target environment, schedule, staffing and probability. Users can modify all inputs to their specifications at any time.

There are five sets of knowledge bases that automatically input environment factors. These knowledge bases cover a wide variety of scenarios and help users produce fast and reliable estimates. Knowledge bases are easily calibrated to user environments to give quick and accurate estimates for the entire life cycle. Users can also change and modify each input at any time. Knowledge bases include the following:

Platform describes the primary operating platform. Platform knowledge bases include avionics, business, ground-based, manned space, unmanned space, shipboard, and more.

Application describes the overall function of the software under estimation. Application knowledge bases include computer-aided design, command & control, database, MIS, office automation, radar, simulation, and more.

Development Method describes the development methods to be used during the development. These knowledge bases include Ada, spiral, prototyping, object oriented design, evolving, traditional waterfall, and more.

Acquisition Method describes the scope and type of project being developed or maintained. These knowledge bases include New Development, Concept Reuse, Design Reuse, Integration Only, Language Conversions, Redocumentation, Reengineering, Modifications, Rehosting, Incremental Builds, Year 2000 Modifications, and more.

Development Standard describes the development documentation, quality, test standards and practices which will be followed during the development. These knowledge bases include commercial, ISO-9000, 2167A, 1703, 1679, 7935A and more.

Processing

SEER-SEM uses proprietary algorithms which are found in the back of the User's Manual. Parameter (input) sensitivities and other insights into the model are also found in the user's documentation. Knowledge bases can be printed out by users. SEER-SEM utilizes an integrated process for risk analysis, including a Monte Carlo simulation.

Outputs

SEER-SEM has almost 30 informative reports and charts covering all aspects of software costs, schedules and risk. The Quick Estimate Report is easily tailored to instantly give the user specific details for trade-off analyses and decision support information. A Detailed Staffing Profile follows SEI suggested staffing categories. Risk reports and graphs based on person months, costs, and schedule are standard features. SEER-SEM gives a minimum schedule output. However, schedules, personnel, and other factors can be changed to give effort and cost tradeoffs.

Calibration

Calibration of SEER-SEM involves the effort to customize input values to more closely reflect particular program development characteristics. Calibration mode allows users to enter project actual effort and schedule. From this information, analysis of the estimated values against the actuals is performed, and suggested calibration factors are provided. Further analysis at higher levels is available to identify trends in technology and calibration factors across multiple projects. Custom knowledge bases may be built to store calibration results which may include parameter settings as wells as calibration adjustment factors.

Life Cycle Considerations

SEER-SEM estimates all elements of the life cycle, beginning with the System Requirements Design phase and ending with software maintenance. SEER-SEM has many features which support Life Cycle Cost Analysis. Total life cycle cost is reported in the Basic Estimate Report, Activity Report, and the Labor Allocation Reports. The Set Reference feature allows for quick analysis of what happens to both development and maintenance costs with the change of any parameter.

Support

SEER-SEM baseline maintenance includes all adaptive, perfective and corrective maintenance. Additionally, you may add annual change rate and growth percents to

anticipate any functional growth or enhancements over the software maintenance period. Enhancements and block upgrades can also be estimated.

Contact

Galorath Incorporated
100 North Sepulveda, Suite 1801
El Segundo, CA 90245
310-414-3222
www.galorath.com

SOFTCOST-OO

SoftCost-OO superseded SoftCost-Ada in 1996 with additional facilities such as C++ and object-oriented (OO) Calibrations. The SoftCost-R model was developed by Don Reifer based on the work of Dr. Robert Tausworthe of the NASA Jet Propulsion Laboratory. SoftCost is now marketed by Resource Calculations, Inc. of Englewood, Colorado. It contains a database of over 1500 data processing, scientific and real-time programs. SoftCost-R is applicable to all types of programs and considers all phases of the software development cycle. The model is available for lease on IBM PC's. A separate model SoftCost-Ada is available to model Ada language and other object-oriented environments.

SoftCost-Ada has been developed to match the new object-oriented and reuse paradigm which are emerging not only in Ada, but also C++ and other object-oriented techniques. It contains a database of over 150 completed projects, primarily Ada.

SoftCost-R Inputs

A key input of SoftCost-R is size, which can either be directly input in SLOC or computed from function points. SoftCost-R uses a more sophisticated sizing model than COCOMO; besides reused code, sizes of modules added or deleted may be included. The other inputs are in four categories like COCOMO. Some SoftCost-R inputs are similar to COCOMO, but many of the more than thirty inputs are unique. Examples of unique inputs are use of peer reviews, customer experience, and degree of standardization. Each input except size requires a rating ranging from "very low" to "extra high", with "nominal" ratings having no effect on effort calculations. SoftCost-R also uses COCOMO inputs to compare the results of SoftCost-R with those of an updated version of COCOMO.

SoftCost-Ada Inputs

In the main, the inputs are the same as SoftCost-R, with some changes to reflect the new paradigm. There is no COCOMO comparison.

Processing

SoftCost-R is not a simple regression model. It uses powerful multivariable differential calculus to develop solutions relying on a probability distribution. This provides the ability for the user to perform "what-if" analysis and look at what would happen to schedule if effort were constrained. Such a capability is not present in COCOMO. SoftCost-R is one of the few models for which the mathematical algorithms are completely described in the user's manual. The SoftCost-R equation is:

PM = P0 * Al * A2 * (SLOC)B
where,
PM = number of person-months,
P0 is a constant factor that may be calibrated,
A 1 is the "Reifer cost factor" which is an exponential product of nine inputs,
A2 is a productivity factor computed from 22 inputs,
B is an exponent which may be calibrated.

The user's manual illustrates values assigned to ratings for all model inputs to help the user understand the effect of each input on effort and schedule.

Outputs

SoftCost-R computes an estimate in person-months of effort and schedule for each project, plus a productivity value. Other outputs include a side-by-side comparison with a recent version of COCOMO, several "what if" analysis options, a resource allocation summary for any of three development methods (traditional waterfall, incremental development, or Ada object-oriented), and schedule outputs for Gantt and PERT charts. SoftCost-Ada output formats are similar, and can interface with project planning tools in the same way.

Calibration

The model contains a calibration file, which contains values for multiple calibration constants and cost drivers. The user may change these values to better describe the user's unique environment, and store alternative calibration and WBS files for different jobs. SoftCost-Ada and SoftCost-R are similar.

Life Cycle Considerations

SoftCost-R contains a separate life cycle model for support costs. In addition to SoftCost-R development inputs, life cycle inputs include annual change traffic, length of support period, a sustaining engineering factor, and economic factors. In addition to annual and total support costs, the life cycle model has optional reports for various staffing options, fixed levels of maintenance, and fixed work force levels. Both SoftCost versions are similar, and use the same staff limited approach to life cycle resource allocation.

Contact

Mr. A.J. (Tony) Collins
Resource Calculations, Inc.
7853 East Arapahoe Court, Suite 2500
Englewood, CO 80112-1361
Telephone: (303) 267-0379
Facsimile: (303) 220-5620

PRICE-S

This model was developed originally by RCA as one of a family of models for hardware and software cost estimation. Developed in 1977 by F. Freiman and Dr. R. Park, it was the first commercially available detailed parametric software cost model to be extensively marketed and used. In 1987, the model was modified and re-validated for modern software development practices. The PRICE-S model is proprietary, it can be leased for yearly use on IBM or compatible PC, and operates within Microsoft windows. It is also available for use on a UNIX workstation. The model is applicable to all types of software projects, and considers all DoD-STD-2167A development phases.

Inputs

One of the primary inputs for the PRICE-S model is source lines of code (SLOC). This may be input by the user or computed using either object-oriented or function point sizing models. Both sizing models are included in the PRICE-S package. Other key inputs include:

1. Application: a measure of the type (or types) of software, described by one of seven categories (mathematical, string manipulation, data storage and retrieval, on-line, real-time, interactive, or operating system).

2. Productivity Factor: A calibratable parameter which relates the software program to the productivity, efficiency/inefficiencies, software development practices and management practices of the development organization.

3. Complexities: Three complexity parameters which relate the project to the expected completion time, based on organizational experience, personnel, development tools, hardware characteristics, and other complicating factors.

4. Platform: the operating environment, in terms of specification, structure and reliability requirements.

5. Utilization: Percentage of hardware memory or processing speed utilized by the software.

6. New Design/New Code: Percentage of new design and new code.

7. Integration (Internal): Effort to integrate various software components together to form an integrated and tested CSCI.

8. Integration (External): Effort to integrate various software CSCI's together to form an integrated and tested software system.

9. Schedule: Software project start and/or end dates.

10. Optional Input Parameters: Financial factors, escalation, risk simulation.

Processing

The PRICE-S algorithms are published in the paper entitled "Central Equations of PRICE S" which is available from PRICE Systems. It states that PRICE-S computes a "weight" of software based on the product of instructions and application inputs. The productivity factor and complexity inputs are very sensitive parameters which affect effort and schedule calculations. Platform is known to be an exponential input; hence, it can be very sensitive. A new weighted design and code value are calculated by the model based on the type or category of instructions. Both new design and code affect schedule and cost calculations. Internal integration input parameters affect the CSCI cost and schedule for integrating and testing the CSCI. The external integration input parameter is used to calculate software to software integration cost and schedule.

Outputs

PRICE-S computes an estimate in person effort (person hours or months). Effort can be converted to cost in dollars or other currency units using financial factor parameters. Software development schedules are calculated for nine DoD-STD-2167A phases: System Concept through Operational Test and Evaluation. Six elements of costs are calculated and reported for each schedule phase: Design Engineering, Programming, Data, Systems Engineering Project Management, Quality Assurance, and Configuration Management. The PRICE-S model also contains several optional outputs including over thirty graphs, Gantt charts, sensitivity matrices, resource expenditure profiles, schedule reports. In addition, Microsoft Project files, spreadsheet files, and risk analysis reports can be generated. The risk analysis report is a Cumulative Probability Distribution and is generated using either Monte Carlo or Latin Hypercube simulation.

Calibration

The PRICE-S model can be run in ECIRP (PRICE backwards) mode to calibrate selected parameters. The most common calibration is that of the productivity factor, which, according to the PRICE-S manual, tends to remain constant for a given organization. It is also possible to calibrate platform, application, and selected internal factors.

Life Cycle Considerations

The PRICE-S life cycle model, included in the PRICE-S package, is a detailed model, which computes software support costs. The primary inputs include PRICE-S development inputs, support descriptors which include software support life, number of installations, expected growth, and support productivity factors. The model also has a modification mode, which allows up to four modifications per software CSCI. The PRICE-S life cycle model calculates support effort and outputs the cost in three support phases: maintenance, enhancements, and growth. The model allocates effort or cost across six elements of costs for each support phase.

Risk Analysis

The PRICE-S model contains a robust Monte Carlo simulation utility, which facilitates rigorous risk analysis. Uncertainty can be characterized using probability distributions to define input parameters. Normal, Beta, Triangular and Uniform distributions are among those available. Simulation results are consolidated and reported as a probabilistic estimate.

Contact
PRICE Systems
700 East Gate Drive, Suite 200
Mt. Laurel, NJ 08054 (800) 437-7423 a.k.a (800) 43PRICE

SoftEst (A Windows version of REVIC)

The Revised Intermediate COCOMO (REVIC) model was developed by Ray Kile and the U.S. Air Force Cost Analysis Agency. It is a copyrighted program that runs under DOS on an IBM PC or compatible computer. The model predicts the development costs for software development from requirements analysis through completion of the software acceptance testing and maintenance costs for fifteen years. REVIC uses the intermediate COCOMO set of equations for calculating the effort (man-power in staff-months and staff-hours) and schedule (elapsed time in calendar months) to complete software development projects based on an estimate of the lines of code to be developed and a description of the development environment. The forms of the basic equations are:

$$MM = AB(KDSI)P(Fi) \quad (1)$$
$$TDEV = CD(MM) \quad (2)$$

Equation (1) predicts the manpower in man-months (MM) based on the estimated lines of code to be developed (KDSI = Delivered Source Instructions in thousands) and the product of a group of environmental factors (Fi). The coefficients (A,C), exponents (B,D) and the factor (Fi) are determined by statistical analysis from a database of completed projects. These variables attempt to account for the variations in the total development environment (such as programmer's capabilities or experience with the hardware or software) that tend to increase or decrease the total

effort and schedule. The results from equation (1) are input to equation (2) to determine the schedule (TDEV = Development Time) in months needed to complete the development.

REVIC enhancement of the intermediate COCOMO includes:

The addition of a fourth mode - Ada. Intermediate COCOMO has three modes of software development: organic, semi-detached, and embedded. These modes describe the overall software development in terms of size, number of interfaces, and complexity. REVIC adds a fourth mode, Ada development, to the model. This mode describes programs developed using an object oriented analysis methodology or use of the Ada language (with separately compilable specifications and body code parts). Each mode provides a different set of coefficients for the basic equations.

Addition of the first and last development phases. COCOMO provides a set of tables distributing the effort and schedule to the phases of development (system engineering, preliminary design, critical design, etc.) and activities (system analysis, coding, test planning, etc.) as a percentage of the total. COCOMO covers four development phases (preliminary design, critical design, code and unit test, and integration and test) in the estimate. REVIC adds two more development phases: software requirements engineering, and integration & test after FQT.

REVIC predicts the effort and schedule in the software requirements engineering phase by taking a percentage of the development phases. It provides a default value (12% for effort, 30% for schedule) for this percentage based on average programs, but allows the user to change the percentage.

COCOMO development phase ends at completion of the integration & test phase (after successful FQT). This phase covers the integration of software CSC's into CSCI's and testing of the CSCI's against the test criteria developed during the program. It does not include the system level integration (commonly called software builds) of CSCI's, and the system-level testing to ensure that system-level specification requirements are met. The software to software and software to hardware integration and testing is accounted for in the Development Test and Evaluation (DT&E) phase. REVIC predicts the effort and schedule for this phase by taking a percentage of the development effort. REVIC provides a default percentage of 22% for effort and 26% for schedule based on average programs. It allows the user to modify these percentages if desired.

Complete interface between the model and the user. Users are not required to have extensive knowledge about the model or detailed knowledge of algorithms. REVIC contains extensive prompting and help screens. REVIC also removes the typical intimidation factor that prevents analysts from successfully using models.

Provide the capability to interactively constrain the schedules and staffing levels. Schedules can be constrained either in the aggregate or by phase of the development effort. Staffing can be constrained by phase. Using these features, the analyst can

estimate cost overruns, underruns, and schedule slips at any major milestone by entering the actuals-to-date at any milestone, and letting the program calculate the remaining effort and schedule.

Inputs: Same as COCOMO inputs.

Processing

While REVIC processing is mostly the same as Intermediate COCOMO, it provides a single weighted "average" distribution for effort and schedule, along with the ability to allow the user to vary the percentages in the system engineering and DT&E phases. On the other hand, COCOMO provides a table for distributing the effort and schedule over the development phases, depending on the size of the code being developed. REVIC has been enhanced by using statistical methods for determining the lines of code to be developed. Low, high, and most probable estimates for each CSC are used to calculate the effective lines of code and standard deviation. The effective lines of code and standard deviation are then used in the equations, rather than the linear sum of the estimates. This quantifies, and to some extent, reduces the existing uncertainties regarding software size. [see Boehm's Software Engineering Economics for a discussion of effective lines of code]. REVIC automatically performs sensitivity analysis showing the plus and minus three sigma values for effort, and the approximate resultant schedule.

Outputs

The user is presented with a menu allowing full exercise of the analytical features and displays of the program. All inputs are easily recalled and changed to facilitate analyses and the user can constrain the solution in a variety of ways.

Calibration

REVIC's coefficients have been calibrated using recently completed DoD projects (development phase only) by using the techniques in Dr. Boehm's book. On the average, the values predicted by the effort and schedule equations in REVIC are higher than in COCOMO. A study validating REVIC equations using a database different from that used for initial calibration was published by the Air Force's HQ AFCMD/EPR. In terms of accuracy, the model compares favorably with expensive commercial models.

The level of accuracy provided by the model is directly proportional to the user's confidence in the lines-of-code (LOC) estimates and a description of the development environment. When little is known about the project or environment, the model can be run leaving all environment parameters at their default (nominal) settings. The only required input is LOC. As the details of the project become known, the data file can be reloaded into the program, and the nominal values can be changed to reflect the new knowledge permitting continual improvement of the accuracy of the model.

Support

REVIC provides an estimate for software maintenance over a fifteen-year period by using the Boehm's COCOMO equation:

$$MMam = MMnom * ACT\ P\ (MFi) \quad (3)$$

where, MMnom is the result of equation (1) without multiplying by the environmental factor (Fi); ACT is annual change traffic as a percentage, and Mfi is the environmental factors for maintenance.

REVIC provides a default percentage of ACT and allows it to be changed. REVIC also assumes a transition period after delivery of the software, during which residual errors are found before reaching a steady state condition. This provides a declining, positive delta to the ACT during the first three years. Beginning the fourth year, REVIC assumes the maintenance activity consists of both error correction and new software enhancements.

Other Reference Materials

There is context-sensitive help available on-line while running REVIC, and the information is enough to input data and obtain results in all cases. However, the developers recommend that Boehm's Software Engineering Economics be read to fully understand the implications of parametric modeling.

Contact

The Air Force Cost Analysis Agency has assumed responsibility for REVIC upkeep and distribution. For suggestions for upgrades or problem reporting, contact:
Air Force Cost Analysis Agency
AFCAA/FM
1111 Jefferson Davis Highway
Crystal Gateway North #403
Arlington, VA 22202
(703) 604-0412

SPR KnowledgePLAN 2.0 (Update to Checkpoint)

The SPR KnowledgePLAN 2.0 was developed by Capers Jones. SPR KnowledgePLAN is marketed by Software Productivity Research of Burlington Massachusetts. KnowledgePLAN 2.0 uses a knowledge database collected from nearly 7,000 software projects from around the world. KnowledgePLAN 2.0 is applicable to all types of programs and considers all phases of the software development cycle. The model is available for sale on IBM compatible PC's.

KnowledgePLAN 2.0 uses function points as the primary means for estimation. SLOC inputs are converted to function points to perform estimation calculations.

Inputs

A key input of KnowledgePLAN 2.0 is size, which can either be directly input by function points or SLOC. If the size of the software is not known, then the project sizing wizard can be used to get an estimate. The main estimation parameters are: classification, complexity, sizing, attributes, goals, calibrate, and task categories. Most parameters are further subdivided with various input tabs. For example, attributes are subdivided into personnel, technology, process, environment, product, and maintenance.

Processing

KnowledgePLAN 2.0 is not a simple regression model. It uses algorithms, rules, and adjustments stored in the project's and SPR's knowledge bases, and task properties. SPR KnowledgePLAN 2.0 has over 100 adjustment variables that may influence an estimate.

Outputs

The SPR KnowledgePLAN 2.0 computes an estimate in person-months of effort, schedule, and cost for each project. Various reports and tables are available, including Gantt charts. The software also allows for easy exportation/importation into other software programs such as Microsoft Project and MPX-Compatible applications.

Calibration

The model can be calibrated at the project or task category levels. Project level refinement occurs through the Calibrate menu and task refinement can be accomplished through changes in task category inputs (project input menu). Major calibration efforts can be accomplished by modifying templates, creating new templates, or with help from SPR to develop a new knowledge base.

Life Cycle Considerations

KnowledgePLAN 2.0 contains a separate database for development and support costs. Separate estimates must be generated for the development and support effort. For the support effort the following areas are addressed: resource loading scheduling, and maintenance attributes. The maintenance attributes are subdivided into personnel, technology, process, environment, and product for a total of 19 inputs.

Contact

Mr. Bob Haven
Software Productivity Research (SPR)
One New England Executive Park
Burlington, MA 01803-5005 USA
Telephone: (781) 273-0140 (http://www.spr.com)
Facsimile: (781) 273-5176

Database Summary of the Models

Table 5. Model Database Legacy

CURRENT DATABASES	REVIC	SASET	SEER-SEM	PRICE-S	SLIM	SoftCost-	Cheek Point
Ada	X	X	X	X	X	X	X
Other Languages Commercial			X	X	X	X	X
Other Languages Government	X	X	X	X			X

(Stukes, 1996)

Summary

This chapter reviewed current issues in the software support area, estimating methodologies, normalization, and the cost models being evaluated. The overall theme of this literature was consistent: software support needs help, and fast, in order to address the software support crisis and operating and support funding requirements. Numerous software support activities have been identified in the literature, however, Table 6 shows that some models lack the identification of various activities, which are performed during the support effort.

Table 6. Software Support Estimate Attribute

SUPPORT ATTRIBUTES	REVIC	SASET	SEER-SEM	PRICE-S	SLIM	SoftCost-OO	CheckPoint
Support Definition includes:							
Maintenance	X	X	X	X	X	X	X
Adaptation to Requirements			X	X			
Enhancements			X	X			X
New Functions (Growth)			X	X			
Support Estimate Based on:							
Factor from Develop. Effort	X	X	X			X	
Unique equation(s)				X	X		X

(Stukes, 1996)

III. Methodology

Overview

This chapter discusses the methodology used for this research effort, which is a follow-on effort to the Coggins and Russell effort of 1993. This effort will concentrate on Software Support, while the Coggins and Russell effort focused on Software Development. This analysis will be conducted in two steps. First, an independent analysis of the selected cost models will be performed using the base test case established in the Coggins and Russell effort (Coggins and Russell, 1993:93). Second, validation will be performed and concurrence will be sought with model vendors and experts in the field.

Independent Analysis (Step 1)

Using the Coggins and Russell base test case, the researchers became familiar with the selected cost estimating models. Appendix A illustrates the Coggins and Russell base case. After gaining familiarity and getting comfortable with using the models, the researchers examined the underlying assumptions and known algorithms of each model to identify significant differences. The researchers also performed additional regression analysis in trying to discover unpublished algorithms within the models for the given relevant range of the data.

The researchers used books, manuals, telephone calls (to experts and technical assistance), personal interviews, and "hands-on" experience to become knowledgeable with how the models operated. A focal point for each of the given models was contacted when additional assistance was required. Telephone interviews were the primary means of communicating with the focal points. Most models had to be used several times before

a "comfort level" was reached with the researchers. After the comfort zone was reached, the researchers turned their attention to the underlying assumptions and algorithms.

The underlying assumptions and algorithms were evaluated to identify significant differences in the software support costs in the models. The team worked together evaluating each model, but also used a checklist to insure all models were evaluated equally. The complete checklist is in Appendix B. An example of some of the questions on the checklist are:

> Issue 1. What are the unique input parameters that directly affect the support costs?
>
> Issue 2. What is the estimating methodology being used?
>
> Issue 3. What are the underlying algorithms?
>
> Issue 4. What is the underlying basis for the parameter value?
>
> Issue 5. Is support broken out into support sub-categories?
>
> Issue 6. What types of sub-categories are included, for example: Preventive, Adaptive, and Corrective?
>
> Issue 7. What time span is the maintenance/support option covering for the various models (5-20)?
> (Parallels Coggins and Russell, 1993:23 & 104-106)

The baseline test case from the Coggins and Russell effort was used for two primary reasons. First, this effort is a follow-on to the Coggins and Russell effort. Secondly, a fictional case was preferred over actual data for numerous reasons. The first was to prevent possible legal issues when the results are published. The second was to examine the means by which the models develop the estimate rather than trying to validate the accuracy of an estimate. This effort examines the model's estimate and how it was derived and not the accuracy of the model to a "real world" situation. This effort

focuses solely on the differences in the estimates provided by the selected models and not the accuracy of the selected models.

Using the same inputs as the Coggins and Russell effort, and nominal inputs for all other areas, software support estimates were determined for each model. The table of inputs and the results are included in Appendices C-G and in Chapter IV-Findings, respectively.

Validation and Concurrence (Step 2)

This phase focused on insuring the researchers had not inadvertently overlooked a key component or input of each model. The steps included:

1) Interviewing model experts and vendors to obtain information not found during the independent analysis.

2) Validating the accuracy of the research results with model experts and vendors.

3) Documenting the results of validations and concurrences with vendors.

4) Reevaluating and updating research after discussions with vendors if required.

(Coggins and Russell, 1993:24)

This phase relied heavily on telephone and personal interviews with the model vendors and experts in the field. There was significant overlap between Steps 1 and 2, since many of the algorithms are not published or readily available, and more guidance was needed at times than what was included in the users' manuals that were provided to the researchers.

IV. Findings

Overview

This chapter provides the results of the model evaluation of the Price-S, SEER-SEM, SoftCost-OO, SoftEst, and SPR KnowledgePLAN. The results of the research effort are summarized below followed by a detailed section on each model that was selected for the comparative study.

The models estimated a twenty-year cost that varied from $35M for PRICE-S to $101M for SEER-SEM with the other models' estimates falling within the range, with the exception of SPR KnowledgePLAN, which does not have a twenty-year maintenance template. Because a fictitious case was used the accuracy of the models was not examined, but the manner in which the models developed the estimates and the differences between the models were addressed. Table 7 is a summary of the results of the model estimates for the first five years of support and the 20-year total.

Table 7. Summary of Model Estimates

Model	PRICE-S	SEER-SEM	SoftCost-OO	SoftEst	SPR KnowledgePLAN
1ST 5Yrs	$14.3M	$36.9M	$15.0M	$25.9M	$20.8M
Full 20Yrs	$34.9M	$101.3M	$57.6M	$91.2M	*
* The default time period for a support effort in KnowledgePLAN 2.0 is a 5-year period.					

PRICE-S

The PRICE-S model gave an estimate of $34.9M for the 20-year maintenance period analyzed. The support costs were categorized or broken out into maintenance, enhancement, and growth. These three categories of support were then further refined into the following five subcategories: design engineering, programming, data systems eng/prog mgmt, quality assurance, and configuration control. The three main support categories were also broken out on a yearly basis for the 20-year duration of the support effort. The model does not directly consider the cost of updating the documentation (change pages) as a separate line item in the model, however it is considered in the data and it is the last calculation made. The PRICE-S model accounts for program management in the systems engineering / program management category. The model also allows for variations in the number of years that the software is to be supported.

The PRICE-S definition of Source Lines Of Code (SLOC) is that it is the total number of lines of code to be developed and or purchased. Comments imbedded in the code are not counted in the number of source lines of code. Type declarations and data statements are included in SLOC and are broken out separately through in the FRAC input parameter.

The first step of the analysis for the PRICE-S model consisted of identifying the segmented parameters that can affect the support effort of software. The first parameter area that was addressed in the analysis was the developmental area. The developmental parameters for CSCI 2 were used as a representation of the affects that parameter changes have on the model and the estimated effort, as well as to keep consistency in the analysis by addressing CSCI 2 for the remaining models when appropriate. The parameters values were first increased in value when possible and then decreased in value when

possible. The results of the variations of the development parameters can be found in Table 8.

Table 8. Variations in Development Parameters for CSCI 2

Baseline for CSCI 2		977.7 Person Months				
Development Parameters for CSCI 2	**Decrease Parameter Value Inputs**	**New Estimate for Person Months**	**Change in Person Months**	**Increase Parameter Value Inputs**	**New Estimate for Person Months**	**Change in Person Months**
Mgmt Complexity	1.0 - 0.9	No Change	0.00%	1.0 - 1.2	No Change	0.00%
External Integration	.50 - .30	No Change	0.00%	.50 - .70	No Change	0.00%
Internal Integration	.50 - .30	968.40	-0.95%	.50 - .70	989.00	1.16%
Utilization	Time .5	----------	0.00%	Time .55	1009.30	3.23%
	Memory .5	----------	0.00%	Memory .55	1009.30	3.23%
Platform	1.8 - 1.4	852.20	-12.84%	1.8 - 2.0	1018.10	4.13%

The first parameter value analyzed was management complexity (CPLXM). Management complexity provides a quantitative description of the relative effect of complicating factors on the overall software task. A value of 1.0 represents an industry average for management complexity. The management complexity parameter was found not to have an effect on the support effort, whether increasing or decreasing the parameter value. The next parameter evaluated was the external integration parameter (INTEGE). External integration is the variable that represents the level of difficulty of integrating and testing the CSCIs to the system level. A value of .5 is typical and a value less than .5 represents more complex tasks. The external integration parameter was found not to have an effect on the support effort by increasing or decreasing in the parameter value. The third parameter analyzed was internal integration (INTEGI). The internal integration parameter represents the level of integrating and testing components to the CSCI level. A typical value is .5, a value less than .5 is a simpler integration effort

and a value greater than .5 is a requirement for more effort. Internal integration showed a modest decrease to 968.4 from 977.7 or a 0.95% decrease, when the value of the parameter was decreased from .50 to .30. When the internal integration value was increased from .50 to .70. The support effort showed an increase in value to 989.0 from 977.7 or a 1.16% increase. Figure 7 represents a graph of the variations of internal integration parameter and the resulting level of effort in Person Months. A linear regression line was then fitted to the three data points with a resulting R-squared value of .9969. The researchers are confident that the regression equation can predict the effort in person months within the given relevant range with 99% of the variation in the predicted value being explained by the regression equation.

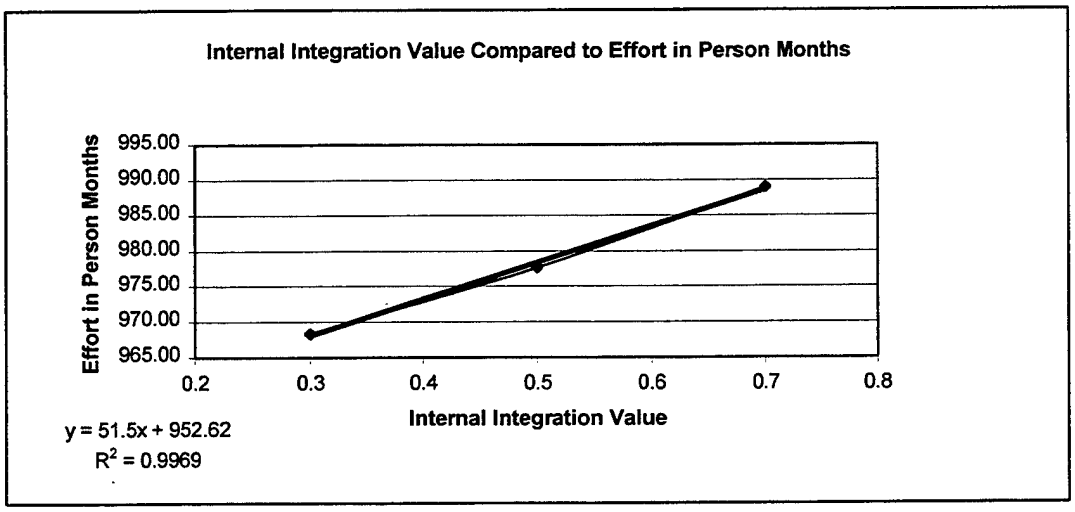
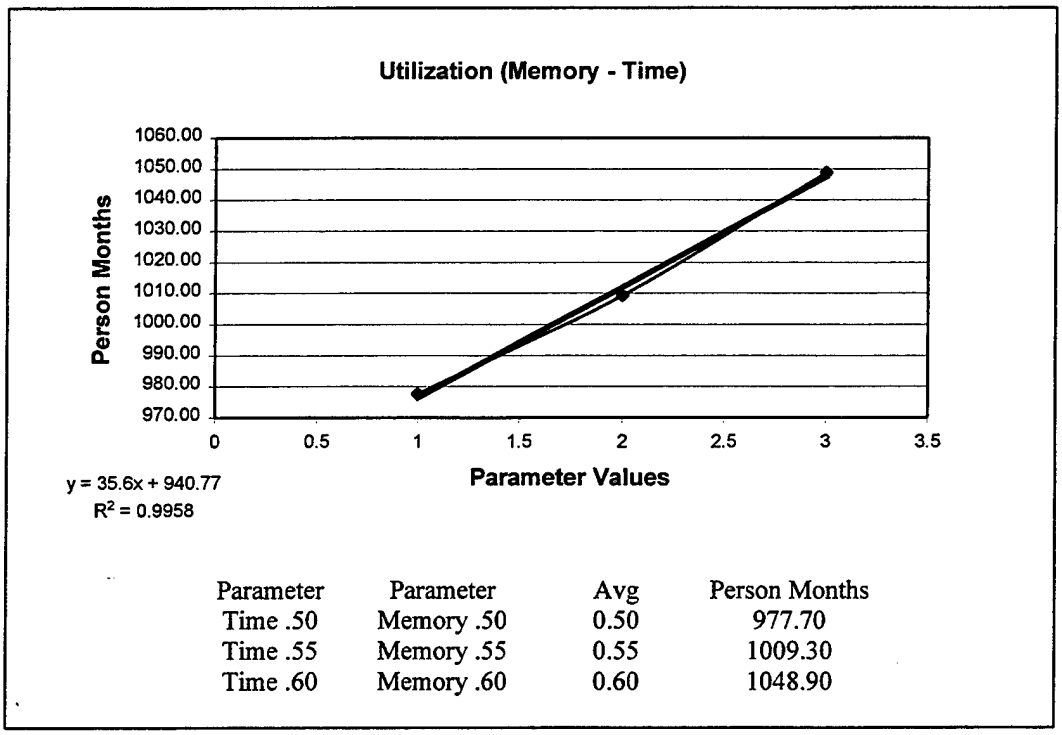


Figure 7. Variations in Internal Integration for CSCI 2 of the Baseline Case

The utilization parameter was the fourth parameter evaluated in the development parameter section of the model. The utilization parameter is subdivided into time and memory. The utilization parameter represents the total cycle time or the total memory capacity used. This variable represents the effort needed to adapt the software to run

within the processor limitations. The initial parameter value of 5.0 could not be decreased or adjusted below the default value of 5.0. Therefore, a series of increases in the utilization parameter were looked at in the analysis. The first incremental increase was made from 5.0 to 5.5, which resulted in an effort of 1009.3 person months or an increase of 3.23% above of the baseline effort. A second increase was made from 5.0 to 6.0, which resulted in an effort of 1048.9 person months or an increase of 7.28% above of the baseline effort. The results and a graph of the variations of the utilization parameter can be found in Figure 8. A linear regression line was then fitted to the three data points with a resulting R-squared value of .9958.



Parameter	Parameter	Avg	Person Months
Time .50	Memory .50	0.50	977.70
Time .55	Memory .55	0.55	1009.30
Time .60	Memory .60	0.60	1048.90

Figure 8. Utilization Variations Compared to Person Months for CSCI 2

The fifth parameter analyzed was platform. The platform parameter was found to have an affect on the support effort by increasing or a decreasing the parameter's value.

A decrease in the platform value from 1.8 (manned air) to 1.4 (mobile) resulted in a decrease in effort to 852.2 person months or a decrease of 12.84% from the baseline case effort. The platform parameter also resulted in an increase in effort to 1018.1 person months or 4.13% when the parameter value was increased from 1.80 (manned air) to 2.0 (unmanned space). Note that the amount of increase and the amount of decrease in the parameter value vary the estimated effort by 4 - 13 % depending on the platform that the software will be used for in the system. The changes in the values were a one unit increase and a one unit decrease from the baseline case used in the study. Figure 9 represents a graph of the variations of the platform parameter and the resulting level of effort in person months.

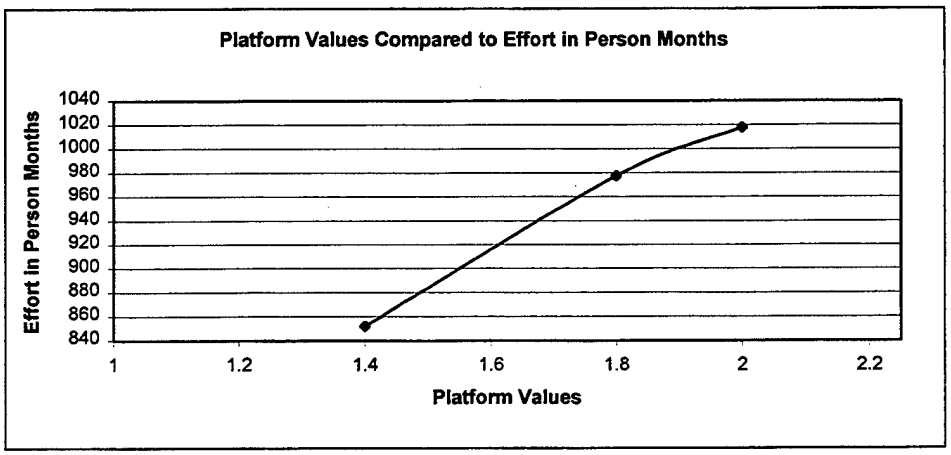


Figure 9. Platform Variations Compared to Person Months for CSCI 2

The next area of input parameters analyzed in the PRICE-S Model was the input section for the language of the software. Within the language section, the following types of inputs are allowed: complexity parameter (CPLX1), hardware software integration complexity parameter (CPLX2), the productivity factor parameter (PROFAC), type of language (LANG), fraction of non-executable code (FRAC), application (APPL), new design (NEWD), new code (NEWC), and SLOC.

Definitions for the Language Parameter Inputs:

Complexity parameter (CPLX1). CPLX1 is a quantitative description of the relative effect of complicating factors on the development effort. Input variables include product familiarity, personnel attributes, software tools, and any unusual factors in the development arena. CPLX1 is directly related to performance schedule. CPLX1 and schedule are treated as a like pair. CPLX1 is used to develop a typical schedule for the estimate. The typical schedule value is compared with the input schedule to assign costs for the excessive acceleration of stretch out. A value of 1 represents an industry average of a normal project/program and with a normal set of programmers.

Hardware Software Integration (HIS) complexity parameter (CPLX2). CPLX2 is a quantitative description of the relative effect of complicating factors on the development effort caused by hardware and software interactions. A value of 1 is an industry average.

Productivity factor parameter (PROFAC). PROFAC is an empirically derived parameter that includes skill levels, experience, productivity, and efficiency. PROFAC provides a way to consolidating, calibrating, and incorporating the net effect of organizational tendencies on the development costs.

Language (LANG). LANG is the language to be used for the development effort

Fraction on non-executable code (FRAC). FRAC is the fraction of source lines of code that represent the data statements and the type declarations.

Application (APPL). APPL is a parameter that summarizes the application of the mix of instructions. APPL has a normal range from 0.866 to 10.952. Values close to the lower end of the continuum represent mathematical strings. Values at the higher end represent real time command and control and interactive applications representing a more difficult coding task.

New design (NEWD). NEWD is the percentage of new and unique design effort to be completed for the development project. NEWD is effort based and not based on SLOC.

New code (NEWC). NEWC is the percentage of new and unique code effort that is effort based and is not based on SLOC.

Some of the language parameters had an influence on the level of effort for supporting the given software. Table 9 shows an overall summary of the affects that the language parameters had on the support effort. As can be seen from Table 9, complexity parameter, hardware software integration complexity parameter, and the production factor parameter did not have an influence on the support effort when the values were

increased and then decreased. However, the application and language parameters did have an affect on the support effort. When the application parameter was decreased from 5.5 to 5.0, a decrease to 897.1 person months or 10.08% decrease in the level of effort was observed. When the application parameter was increased from 5.5 to 6.0 an increase in the level of effort to 1078.9 or a 10.35% increase was observed. The LANG parameter in the language input environment of the model had the largest affect on the support effort estimate. A change from C++ to Ada95 resulted in an increase in effort to 1145.3 person months or a 17% increase in the support effort, thus accounting for language differences.

Table 9. Variations in Language Parameters for CSCI 2

Baseline for CSCI 2		**977.7 Person Months**				
Language Parameters for CSCI 2	**Decrease Parameter Value Inputs**	**New Estimate for Person Months**	**Change in Person Months**	**Increase Parameter Value Inputs**	**New Estimate for Person Months**	**Change in Person Months**
Complx 1	0.8 – 0.7	No Change	0.00%	0.8 - 1.0	No Change	0.00%
Complx 2	1.0 – 0.9	No Change	0.00%	1.0 - 1.1	No Change	0.00%
Prod Factor	5.0 – 4.0	No Change	0.00%	5.0 - 6.0	No Change	0.00%
Application	5.5 –5.0	879.10	-10.08%	5.5 - 6.0	1078.90	10.35%
Language	C++ - Ada95	1145.30	17.14%			

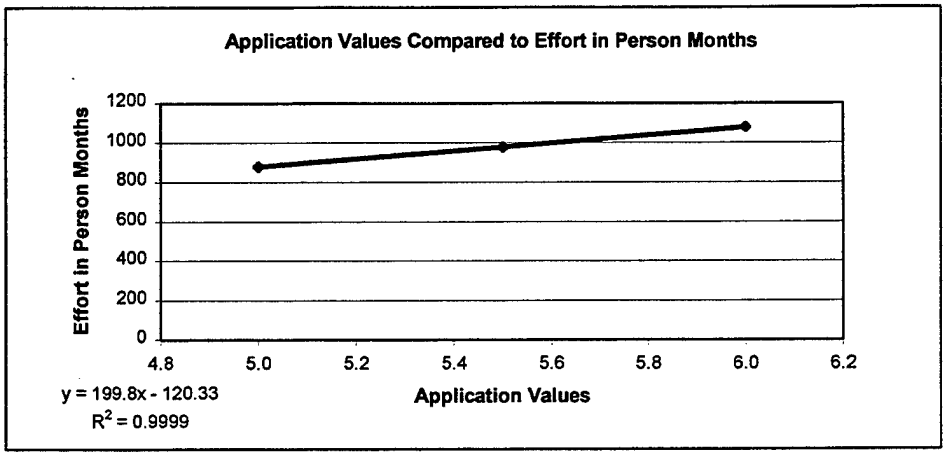


Figure 10. Application Value Variations Compared to Person Months for CSCI 2

PRICE-S also has a separate input section for the support environment labeled deployment section. Within the deployment section, the following types of inputs are allowed: Start and End Date, number of installations, quality level (Q-level), growth level (G-level), enhancement level (E-Level), enhancements factor (EPROFAC), maintenance factor (MPROFAC), growth factor (GPROFAC), and Source Lines of Code (SLOC).

Definitions for the Deployment Parameter Inputs (PRICE-S, 1993):

Quality Level (Q-Level). An input value that adjusts the models estimate for initial defects, which allows the user of the model to adjust maintenance costs. A typical value for Q-Level is 1. A value of 2 would imply that that the delivered code has half the number of errors that PRICE S would compute for the project.

Growth. Growth is the addition of new software functions or capabilities not called for in the original development effort. Growth is represented by the number of executable and non-executable instructions.

Growth Level (G-Level). The increase in the software's size SLOC/KDSI that is anticipated to be added for modifications or new software functions that were not found in the original development effort. A value of 1.5 would represent an increase of 150% of the initial program size over the life of the support effort.

Enhancement. The improvement made in the portability, efficiency, and performance of the software. The purpose of enhancement is to modify the original performance specifications without adding new functions to the software. It also includes those activities that are required to upgrade or modify hardware components. Enhancement is a replacement operation that can often be represented by increases in speed, response times, or through put.

Enhancement Level (E-level). Enhancement level is a modifier to the internally estimated level of enhancement to the code. A typical value for E-Level is normally 1.

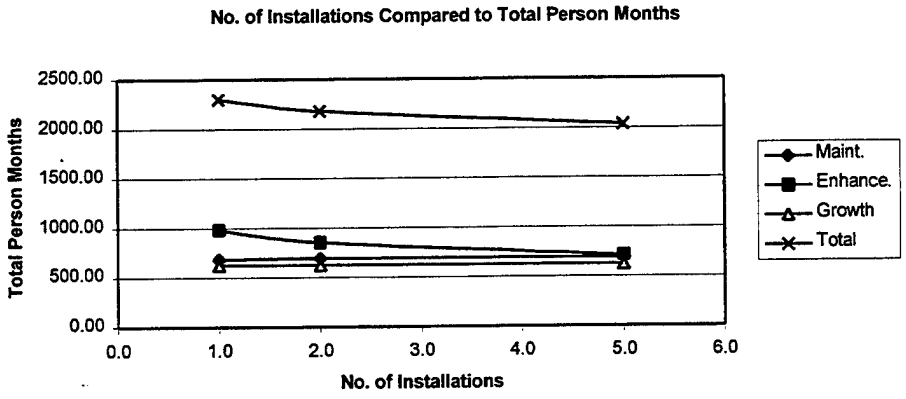
Enhancement Level (EPROFAC). EPROFAC is an empirically derived parameter that includes variables such as skill level, experience, productivity, and efficiency. EPROFAC is often less than the development activity productivity factor.

Maintenance. Maintenance is the effort performed to correct latent defects in the software that prevent it from meeting its original specifications. Maintenance is geared towards fixing bugs in the software. It also includes efforts aimed at making future repairs and system updates more efficient, such as readability and maintainability.

Maintenance PROFAC (MPROFAC). MPROFAC is an empirically calculated parameter that includes skill levels, experience, productivity, and efficiency as related to the maintenance activity. MPROFAC is a mandatory input.

Growth PROFAC (GPROFAC). GPROFAC is a derived parameter that which includes skill level, experience, productivity, and efficiency as related to the growth activity.

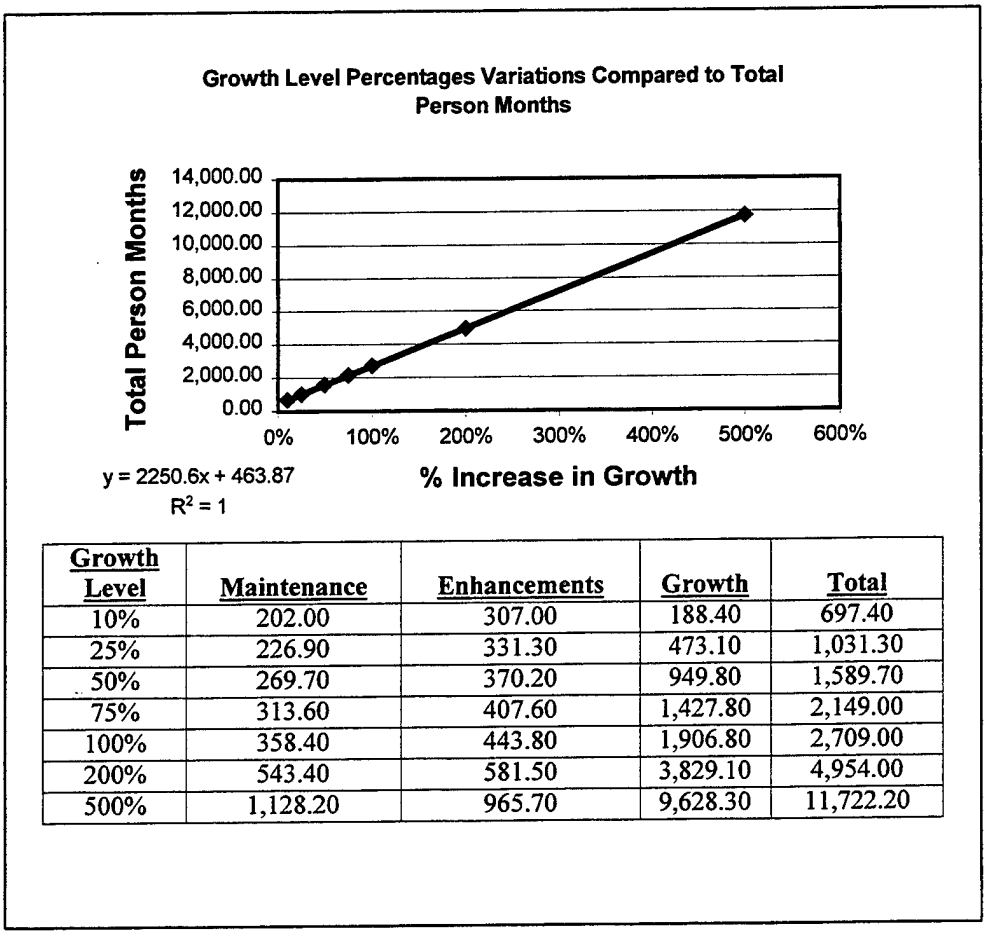
The first parameter analyzed in the deployment input environment was activity installations. Activity installations identifies the maximum number installations where the software is to be deployed. When the value of the parameter was changed from 1 to 2, a 5% decrease in the total support cost was identified with the maintenance category increasing by 1 percent and the enhancement category decreasing by 13%. When the value was increased from 1 to 5, a decrease of 11% was identified with a 2% increase in the maintenance category and a 28% decrease the enhancement category. The larger the number of installations the less enhancements. See figure 11 for installation effort graph.



Parameter	Values	Maint.	Change	Enhance.	Change	Growth	Change	Total	Change
Installations Baseline	1.0	685.40	0.0%	984.50	0.0%	629.20	0.0%	2,299.10	0.0%
Installations Increase	2.0	695.60	1.0%	856.70	-13.0%	629.20	0.0%	2,181.50	-5.0%
Installations Increase	5.0	698.20	2.0%	712.90	-28.0%	629.20	0.0%	2,040.30	-11.0%

Figure 11. No. of Installation Variations Compared to Total Person Months

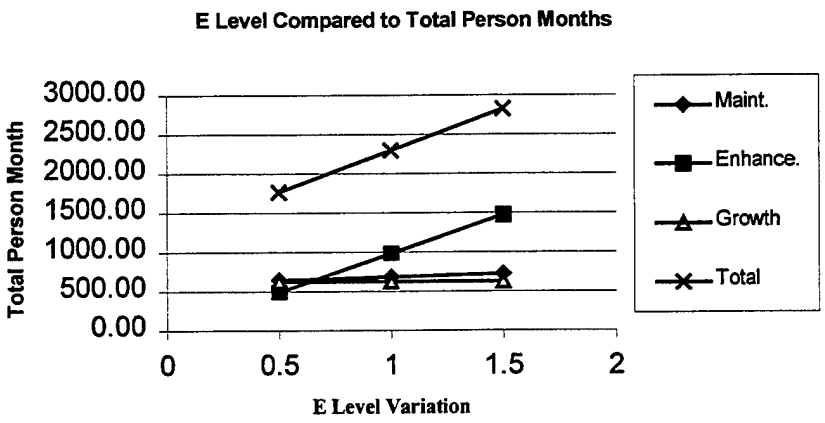
The second deployment parameter examined was Growth Level (G-Level). G-level is the increase in software size SLOC/KDSI that is anticipated to be added for modifications. When the value of the G-Level parameter was changed from 1 to 2, a 5% decrease in the total support cost was identified with the maintenance category increasing by 1 percent and the enhancement category decreasing by 13%. When the value was increased from 1 to 5, a decrease of 11% was identified with a 2% increase in the maintenance category and a 28% decrease the enhancement category. G-Level is a linear multiplier to initial defects. See figure 12 for growth level percentages variations graph.



Growth Level	Maintenance	Enhancements	Growth	Total
10%	202.00	307.00	188.40	697.40
25%	226.90	331.30	473.10	1,031.30
50%	269.70	370.20	949.80	1,589.70
75%	313.60	407.60	1,427.80	2,149.00
100%	358.40	443.80	1,906.80	2,709.00
200%	543.40	581.50	3,829.10	4,954.00
500%	1,128.20	965.70	9,628.30	11,722.20

Figure 12. Growth Level Percentage Variations Compared to Total Person Months

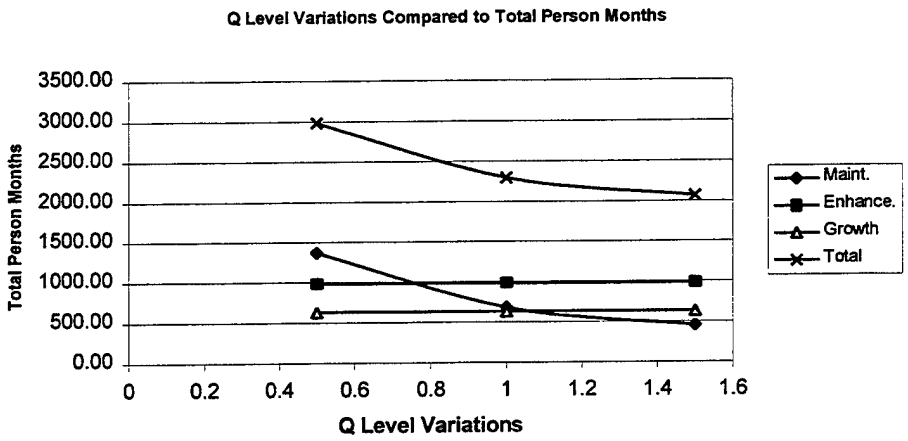
The third deployment parameter examined was Enhancement Level (E-Level). E-level is a modifier to the internally estimated level of enhancement to the code. When the value of the E-Level parameter was decreased from 1 to .5, a 23% decrease in the total support effort was identified with the maintenance category increasing by .06 percent and the enhancement category decreasing by 50%. When the value was increased from 1 to 1.5, a decrease of 23% was identified with a .05% increase in the maintenance category and a 50% increase the enhancement category. E-Level is a linear multiplier to initial defects. See figure 13 for the enhancement level variations graph.



Parameter	Values	Maint.	Change	Enhance.	Change	Growth	Change	Total	Change
E-Level Decrease	0.5	647.40	-0.06%	491.40	-0.50%	629.20	0.00%	1,768.00	-0.23%
E-Level Baseline	1.0	685.40	0.00%	984.50	0.00%	629.20	0.00%	2,299.10	0.00%
E-Level Increase	1.5	722.50	0.05%	1478.30	0.50%	629.20	0.00%	2,830.00	0.23%

Figure 13. E-Level Variations Compared to Total Person Months

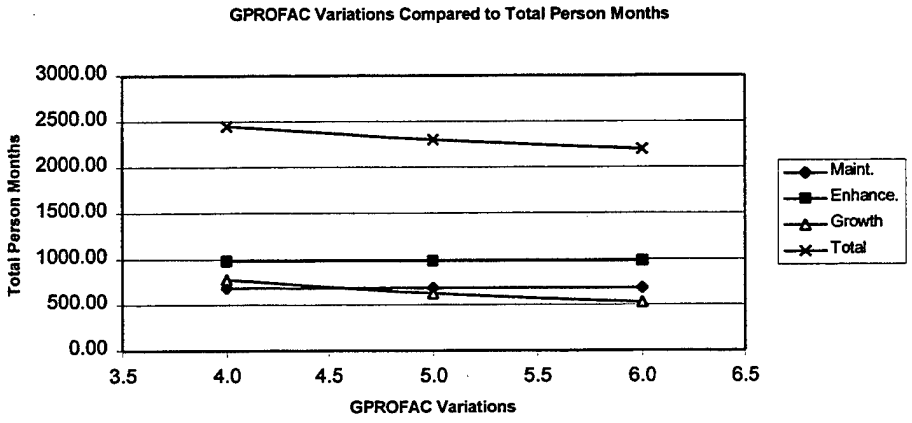
The next deployment parameter examined was Quality Level (Q-Level). Q-Level is an input value that adjusts the models estimate for initial defects. When the value of the Q-Level parameter was decreased from 1 to .5, a 30% decrease in the total support effort was identified with the maintenance category increasing by 100 percent with the enhancement and growth categories remaining unchanged. When the Q-level value was increased from 1 to 1.5, a decrease of 10% in the total support effort was identified with a 33% decrease in the maintenance category with the enhancement and growth categories remaining unchanged. Q-Level is a linear multiplier to initial defects. See figure 14 for quality level variations graph.



Parameter	Values	Maint.	Change	Enhance.	Change	Growth	Change	Total	Change
Q-Level Decrease	0.5	1370.00	100.0%	984.50	0.0%	629.20	00.0%	2,983.70	30.0%
Q-Level Baseline	1.0	685.40	0.00%	984.50	0.0%	629.20	00.0%	2,299.10	0.0%
Q-Level Increase	1.5	457.00	- 33.0%	984.50	0.0%	629.20	00.0%	2,070.70	-10.0%

Figure 14. Q-Level Variations Compared to Total Person Months

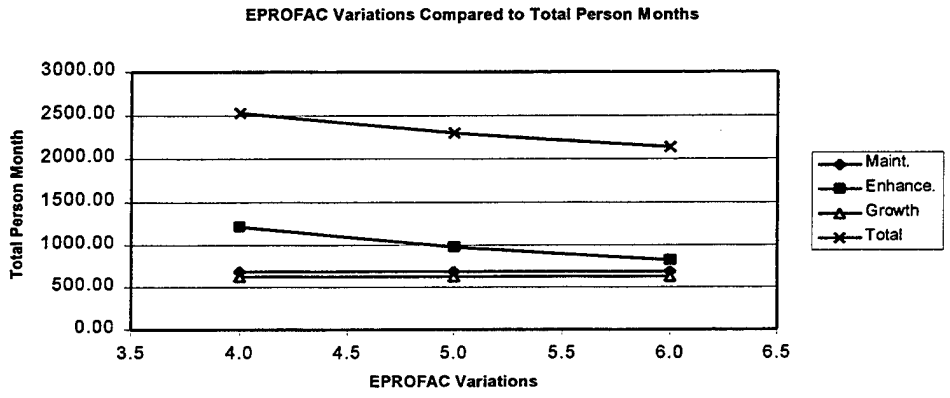
The fifth parameter analyzed was Growth factor. GPROFAC is a derived parameter that includes skill levels, experience, productivity, and efficiency. When the value of the parameter was changed from 5 to 4, a 6 % increase in the total support effort was identified with the growth category increasing by 24% and the enhancement and maintenance categories remaining unchanged. When the value was increased from 5 to 6, a decrease of 4% in the total support effort was identified with a 16% decrease in the growth category and no change to the maintenance and enhancement categories. See figure 15 for GPROFAC variation graph.



Parameter	Values	Maint.	Change	Enhance.	Change	Growth	Change	Total	Change
GPROFAC Decrease	4.0	685.40	0.0%	984.50	0.0%	777.30	24.0%	2,447.20	6.0%
GPROFAC Baseline	5.0	685.40	0.0%	984.50	0.0%	629.20	0.0%	2,299.10	0.0%
GPROFAC Increase	6.0	685.40	0.0%	984.50	0.0%	527.00	-16.0%	2,196.90	-4.0%

Figure 15. GPROFAC Variations Compared to Total Person Months

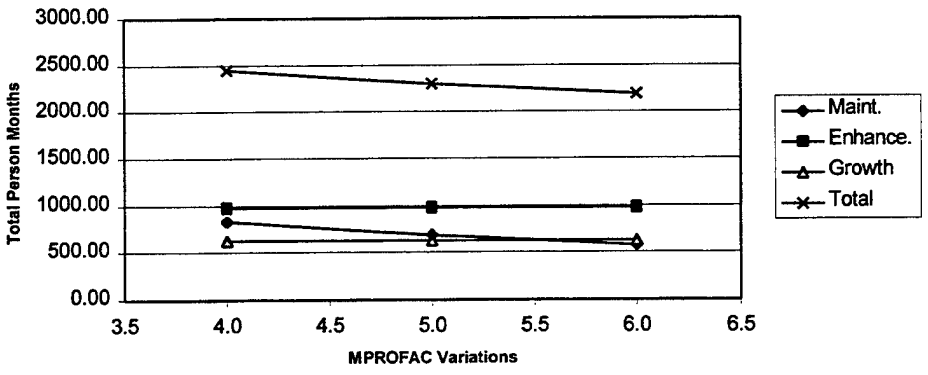
The sixth parameter analyzed was Enhancements PROFAC (EPROFAC). EPROFAC is a derived parameter that includes skill levels, experience, and productivity as related to the enhancement activity. When the value of the parameter was changed from 5 to 4 a 10 % increase in the total support effort was identified with the enhancement category increasing by 23 percent and the growth and maintenance categories remaining unchanged. When the EPROFAC parameter value was increased from 5 to 6 a total support effort decrease of 7% was identified with a 16% decrease in the enhancement category and with no change occurring in the maintenance and growth categories. See figure 16 for EPROFAC variation graph.



Parameter	Values	Maint.	Change	Enhance.	Change	Growth	Change	Total	Change
EPROFAC Decrease	4.0	685.40	0.0%	1214.80	23.0%	629.20	0.0%	2,529.40	10.0%
EPROFAC Baseline	5.0	685.40	0.0%	984.50	0.0%	629.20	0.0%	2,299.10	0.0%
EPROFAC Increase	6.0	685.40	0.0%	825.30	-16.0%	629.20	0.0%	2,139.90	-7.0%

Figure 16. EPROFAC Variations Compared to Total Person Months

The last parameter analyzed was Maintenance PROFAC (MPROFAC). MPROFAC is derived parameter that includes skill levels, experience, and productivity as related to the maintenance activity. When the value of the MPROFAC parameter was changed from 5 to 4, a 7 % increase in the total support effort was identified with the maintenance category increasing by 22 percent and the growth and enhancement categories remaining unchanged. When the value of MPROFAC was increased from 5 to 6, a decrease of 5% was identified in the total support effort with a 15% decrease in the maintenance category and no change to the enhancement and growth categories. See figure 17 for MPROFAC variation graph.



Parameter	Values	Maint.	Change	Enhance.	Change	Growth	Change	Total	Change
MPROFAC Decrease	4.0	836.30	22.0%	984.50	0.0%	629.20	0.0%	2,450.00	7.0%
MPROFAC Baseline	5.0	685.40	0.0%	984.50	0.0%	629.20	0.0%	2,299.10	0.0%
MPROFAC Increase	6.0	579.90	-15.0%	984.50	0.0%	629.20	0.0%	2,193.60	-5.%

Figure 17. MPROFAC Variations Compared to Total Person Months

The PRICE-S model output gives maintenance, enhancement, and growth as the main support effort categories, refer to table 10 for the support effort in person months and table 11 for the total support effort in dollars.

Table 10. Support Effort in Person Months Broken out by Categories and Year

PRICE-S Total System Support Costs for Twenty Years				
Annual Costs in Person Months				
YEAR	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
1998	103.50	22.10	8.70	134.30
1999	116.70	57.50	23.80	198.00
2000	85.40	81.40	35.70	202.50
2001	62.40	95.90	44.50	202.80
2002	47.90	102.30	50.70	200.90
2003	38.70	102.60	54.20	195.50
2004	32.60	98.10	55.50	186.20
2005	28.20	89.90	55.10	173.20
2006	24.90	79.60	52.80	157.30
2007	22.20	67.90	49.40	139.50
2008	20.00	55.80	44.70	120.50
2009	18.00	43.80	39.30	101.10
2010	16.10	33.00	33.40	82.50
2011	14.20	23.40	27.20	64.80
2012	12.60	15.30	21.00	48.90
2013	10.90	9.10	15.20	35.20
2014	9.50	4.60	9.80	23.90
2015	8.20	1.90	5.50	15.60
2016	7.00	0.50	2.30	9.80
2017	5.90	0.00	0.40	6.30
2018	0.40	0.00	0.00	0.40
TOTAL:	685.30	984.70	629.20	2299.20

Table 11. Total Support Effort Represented in Dollars

Total Person Months		Labor rate		Hours Month		Total Cost 20 Yrs
2299.20	*	$100	*	152	=	$34,947,840

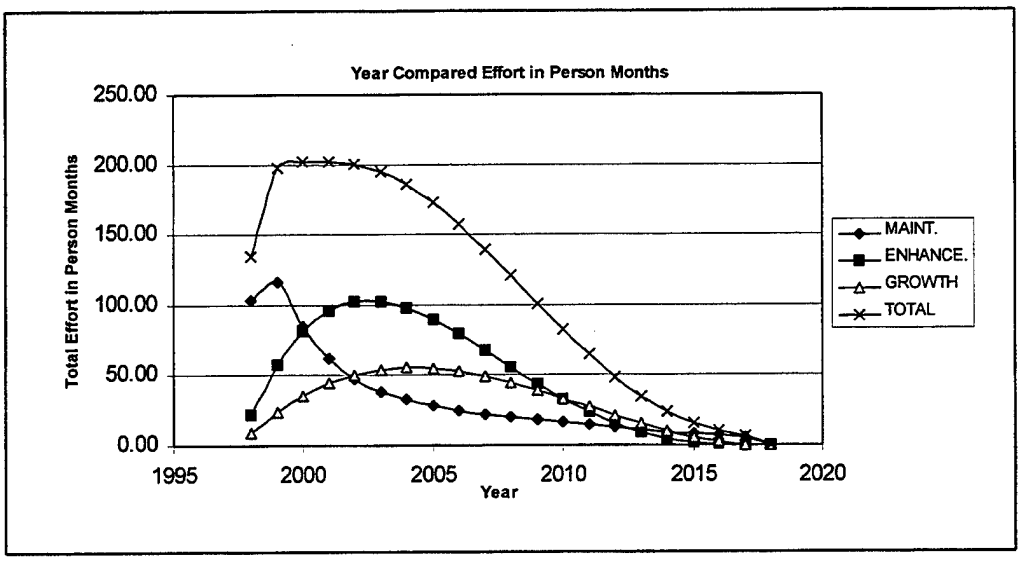


Figure 18. Year Compared to Category Effort in Person Months

The total support effort increases for the first four years and then begins to decrease over the duration of the support effort. The maintenance category increases for the first two years and then begins to decline for the remainder of the support effort. The enhancement category increases for the first six years and then decreases for the remainder of the support effort. The last category growth increases for the first seven years and then decreases over the remainder of the support effort.

Upon interviewing Jim Otte at PRICE Systems, the following parameters within the model where identified as the basis for the maintenance effort. The first parameter was the program size or the SLOC of the software program. The second parameter was application (APPL) with a normal range from .866 to 10.952. The larger the value for APPL the more difficult the programming task in nature. APPL represents an inherent instruction complexity, independent of the variations in schedule, operating environment and system utilization. As the value for APPL goes up the propensity for errors goes up as well. The third parameter was platform, which had a normal the range of 0.6 to 2.5. PLTFM describes the customer's planned operating environment requirements. It is a

measure of the portability, reliability, structuring, testing and documentation required for acceptable contract performance. The key to selecting the appropriate value for PLTFM is not the location of software, but the specification and testing which the software must be meet. Costs and schedules for software development are strongly driven by the severity of the requirements, which must be satisfied for acceptable contract performance. The following options/values can be found under PLTFM for military software: Ground Environment 1.2, Mobile (Van Or Shipboard) 1.4, Airborne 1.8. The second category under PLTFM related to military software is Space software with the following options/values being found under PLTFM: Unmanned 2.0 and Manned 2.5. Space software has higher parameter values due to the inherent risk need to demonstrate extremely high reliability. The consequences of failure are extremely severe with space software due to the lives involved. The only permissible failure mode is an alternate system. Software failure could potentially cost personnel, the mission, and/or the program/project. As the value for PLTFM goes up, software testing is performed more thoroughly for military and space software.

The fourth parameter value was utilization. UTIL is the fraction of available hardware cycle time or total memory capacity used by the software program. It describes the extra effort needed to adapt software to operate within limited processor capabilities. A UTIL value of .5 or less represents normal problems and a value higher than .5 represents problems that are not ordinary. The UTIL dialog box allows you to enter both timing and memory utilization and then computes a composite utilization fraction using the following equation:

$$(\text{UTIL}_{\text{timing}})\,(0.955 - \text{UTIL}_{\text{memory}}) + (0.95)(\text{UTIL}_{\text{memory}}) - 0.475 \qquad (1)$$

The compression of the software schedule does have an effect on the model estimate of the support effort. If the schedule is compressed then the PRICE-S model will estimate more defects in the support effort. The researchers were given additional information to consider from Mr. Otte on how to handle situations in which the software is developed by a different organization than the organization that will be handling the post deployment support effort. If a government organization will support the software then the development PROFAC should be lower causing an increase in the support effort/cost. If the development and support are handled by the same organization then no adjustment is needed. If the support contractor is included in integration and testing then PROFAC should be increased by approximately .5 to account for the contractor assistance.

The database used in the PRICE-S model is comprised of four hundred and fifteen CSCI's and Projects. It includes application in a commercial, military, and a space environment. The platforms range from management information systems (MIS), to manned space with everything from avionics to ship/mobile in between. The database is also categorized into the 7 different language categories. The seven categories include Ada, FORTRAN, C++, JOVIAL, Assembly, 4GL, and other languages. The total SLOC for each language is given as well as the percentage of software program found in a given platform. Table 12 provides a summary of database and the category breakouts. The data represented in the table are from 1992. Therefore, the database composition has more than likely changed since the given time frame, but it is represented here to give the user of the model an idea of the types of software programs that are in included in the database.

Table 12. PRICE-S Database Composition

APPLICATION							
Commercial	Ada	FORTRAN	C++	JOVIAL	ASSY	4GL	Other
Internal	0%	1%	0%	0%	0%	9%	6%
MIS	0%	0%	80%	0%	19%	86%	16%
Avionics	5%	13%	0%	5%	10%	0%	9%
Military							
Ground	9%	10%	16%	0%	10%	4%	14%
Ship/Mobile	24%	34%	0%	10%	20%	0%	26%
Airborne	48%	8%	0%	49%	26%	0%	16%
Missile	4%	3%	0%	10%	4%	0%	4%
Unmanned Space	10%	3%	0%	5%	1%	0%	1%
Manned Space	0%	25%	0%	21%	9%	0%	7%
Other	0%	2%	4%	0%	1%	1%	1%
Total SLOC	5025K	5836K	938K	7775K	6800K	2040 K	7820K

ADA	1984	- Current	Age 1975 to 1992
FORTRAN	1975	- Current	Number of CSCI/Projects 45
C++	1989	- Current	Note: SLOC range per software package was 1,000 – 2,000,000
JOVIAL	1975	- Current	
ASSY	1975	- Current	
4GL	1979	- Current	
Other	1979	- Current	

SEER-SEM

SEER-SEM estimates staffing, effort months of maintenance sub-divided into categories, and cost. The specific cost elements are: Average Staff Level, Corrective Maintenance, Adaptive Maintenance, Perfective Maintenance, and Enhancement Maintenance. The cost elements are summed and a total per year and a cumulative total for both effort and cost are provided. Average staff level refers to the number of people required to maintain the software in a given year. Corrective maintenance is the effort spent correcting software problems, fixing bugs, and addressing design flaws. Adaptive maintenance is the effort spent adapting the software to the current environment, including updating to the latest operating system, compilers, hardware devices, etc. Perfective maintenance is the effort spent fine tuning and perfecting the software, including adjusting control or data files, refining performance issues, and other finishing touches. Enhancement maintenance is the effort to add minor enhancements; major re-engineering is typically estimated as a redevelopment.

No specific line item for program management or updating the documentation is provided in the support report or inputs, although a labor category report does show the breakouts. A source line of code (SLOC) is defined as a non-comment software source instruction. Included in SLOC is executable source lines, control, mathematical, conditional, input/output formatting, data declarations, and deliverable job control language (JCL). Comments, blanks, begin statements, separate physical lines for convenience, machine/library generated statements, non-final test code, and debug statements do not count as a SLOC.

The model does have an input for various software languages, but no changes to the support cost were found by the researchers (refer to Table 13), however, some languages require more lines of code to deliver the same functionality.

Table 13. SEER-SEM Changes from Baseline

SEER-SEM Baseline	**44,596,872**			
	Decrease	Change	Increase	Change
LINES				
Function Implementation Mechanism		N/A		N/A
CSCIs Included In Size		N/A		N/A
PERSONNEL CAPABILITIES & EXPERIENCE				
Analyst Capabilities	47,309,841	6.08%	42,379,814	-4.97%
Analyst's Application Experience	47,369,418	6.22%	42,461,736	-4.79%
Programmer Capabilities	47,453,878	6.41%	42,492,465	-4.72%
Programmer's Language Experience	44,820,457	0.50%	44,495,542	-0.23%
Host System Experience	45,025,969	0.96%	44,430,511	-0.37%
Target System Experience	44,890,239	0.66%	44,492,085	-0.23%
Practices & Methods Experience	44,596,872	0.00%	44,596,872	0.00%
DEVELOPMENT SUPPORT ENVIRONMENT				
Modern Development Practices Use	47,376,293	6.23%	42,203,070	-5.37%
Automated Tools Use	45,940,301	3.01%	43,252,167	-3.02%
Logon thru Hardcopy Turnaround	43,354,588	-2.79%	46,073,614	3.31%
Terminal Response Time	44,173,638	-0.95%	45,301,982	1.58%
Multiple Site Development	44,596,872	0.00%	45,669,327	2.40%
Resource Dedication	46,745,995	4.82%	44,596,872	0.00%
Resource and Support Location	44,596,872	0.00%	46,745,995	4.82%
Host System Volatility	43,598,448	-2.24%	45,599,037	2.25%
Process Volatility	43,305,277	-2.90%	45,721,327	2.52%
PRODUCT DEVELOPMENT REQUIREMENTS				
Requirements Volatility (Change)	43,294,803	-2.92%	47,219,993	5.88%
Specification Level – Reliability	43,058,845	-3.45%	45,734,168	2.55%
Test Level	44,308,319	-0.65%	44,798,170	0.45%
Quality Assurance Level	44,376,840	-0.49%	44,705,552	0.24%
Rehost from Development to Target	40,202,704	-9.85%	48,996,283	9.86%
PRODUCT REUSABILITY REQUIREMENTS				
Reusability Level Required	44,596,872	0.00%	46,207,138	3.61%
Software Impacted by Reuse				
DEVELOPMENT ENVIRONMENT COMPLEXITY				
Language Type (complexity)	44,456,369	-0.32%	44,774,951	0.40%
Host Development System Complexity	44,383,698	-0.48%	44,887,651	0.65%
Application Class Complexity	41,841,012	-6.18%	45,140,964	1.22%
Process Improvement	44,596,872	0.00%	44,596,872	0.00%

Table 13. SEER-SEM Changes from Baseline (Continued)

TARGET ENVIRONMENT				
Special Display Requirements	42,727,587	-4.19%	45,835,723	2.78%
Memory Constraints	43,744,711	-1.91%	46,137,601	3.45%
Time Constraints	43,217,177	-3.09%	46,505,601	4.28%
Real Time Code	43,307,352	-2.89%	46,152,530	3.49%
Target System Complexity	44,459,349	-0.31%	44,776,110	0.40%
Target System Volatility	43,714,437	-1.98%	44,562,858	-0.08%
Security Requirements	44,596,872	0.00%	53,842,775	20.73%
SCHEDULE & STAFFING CONSIDERATIONS				
Required Schedule (Calendar Mos)		N/A		N/A
Start Date		N/A		N/A
Complexity (Staffing)	42,795,787	-4.04%	46,730,624	4.78%
Staff Loading		N/A		N/A
RISK ANALYSIS		N/A		N/A
REQUIREMENTS		N/A		N/A
SYSTEM INTEGRATION		N/A		N/A
ECONOMIC FACTORS		N/A		N/A
SOFTWARE MAINTENANCE				
Years of Maintenance		N/A		N/A
Separate Sites	See Figure 19			
Maintenance Growth Over Life	See Figure 20			
Personnel Differences	46,265,226	3.74%	43,007,376	-3.56%
Development Environment Differences	46,350,551	3.93%	42,988,222	-3.61%
Annual Change Rate	See Figure 21			
Maintenance Level (Rigor)	42,924,490	-3.75%	48,499,099	8.75%
Min Maintenance Staff (Optional)		N/A		N/A
Max Maintenance Staff (Optional)		N/A		N/A
Maintenance Monthly Labor Rate		N/A		N/A
Additional Annual Maintenance Cost		N/A		N/A
Maintenance Start Date		N/A		N/A
Percent To Be Maintained	See Figure 22			
Maintain Total System		N/A		N/A
SOFTWARE CODE METRICS (Optional)				
ESTIMATE TO COMPLETE				
ADJUSTMENT FACTORS				
	No Knowledge		Ada Development	
Development Method	56,589,253	26.89%	60,307,753	35.23%

Note: N/A means this parameter was not addressed during the evaluation. An increase or decrease was determined by a one unit incremental shift in the parameter. An example of a decrease in a parameter input would be a Low, Nom, and Hi would be changed to a Low -, Nom-, and Hi- for the parameter input evaluation. See Appendix C for the baseline inputs.

Model development factors may have a significant impact on the support cost. Please refer to Table 13. A change in Security from unclassified to classified increased the support cost by over 20%. A change in the Development Method from Waterfall to Ada Development also showed a dramatic increase in support cost, over 35% (Table13). Figures 23 and 24 show the support effort by category and year.

Software Maintenance unique inputs are:

Years of Maintenance
Separate Sites
Maintenance Growth over Life
Personnel Differences
Development Environment Differences
Annual change Rate
Maintenance Level (Rigor)
Min Maintenance Staff (Optional)
Max Maintenance Staff (Optional)
Maintenance Monthly Labor Rate
Additional Annual Maintenance Cost
Maintenance Start Date
Percent to be Maintained
Maintain Total System

These are defined as follows:

Years of Maintenance. the number of years for which support will be estimated. Maintenance/Support begins when operational test and evaluation is complete.

Separate Sites. The number of operational sites where software will be installed and the users will have input into enhancements.

Maintenance Growth over Life. The anticipated size growth from the point immediately after software is turned over to maintenance to the end of support.

Personnel Differences. Comparison of the development and maintenance personnel's capabilities and experience.

Development Environment Differences. Rates the quality of the maintenance environment in comparison to the development environment.

Annual change Rate. Average percentage of the software impacted by software maintenance and sustaining engineering per year.

Maintenance Level (Rigor). Rates the thoroughness with which maintenance will be performed.

Min Maintenance Staff (Optional). Minimum number of personnel who will be assigned to support the software.

Max Maintenance Staff (Optional). Maximum number of personnel who will be assigned to support the software.

Maintenance Monthly Labor Rate. Average monthly labor rate for support personnel.

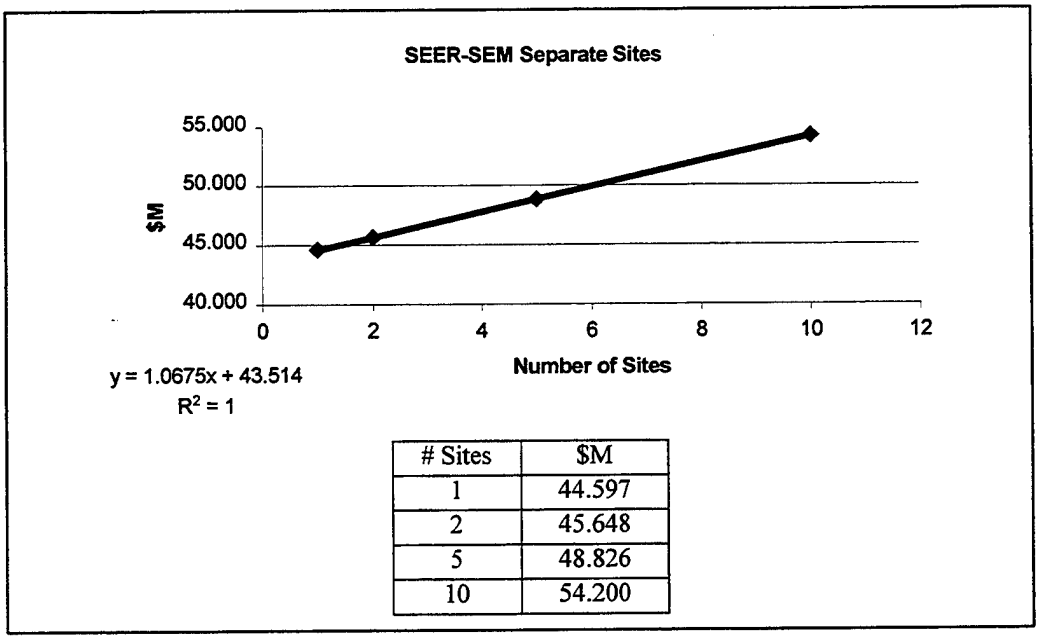
Additional Annual Maintenance Cost. Any annual throughput cost for support

Maintenance Start Date. Date on which support will begin. Default is when operational test and evaluation is completed.

Percent to be Maintained. The percentage of the total that will be supported.

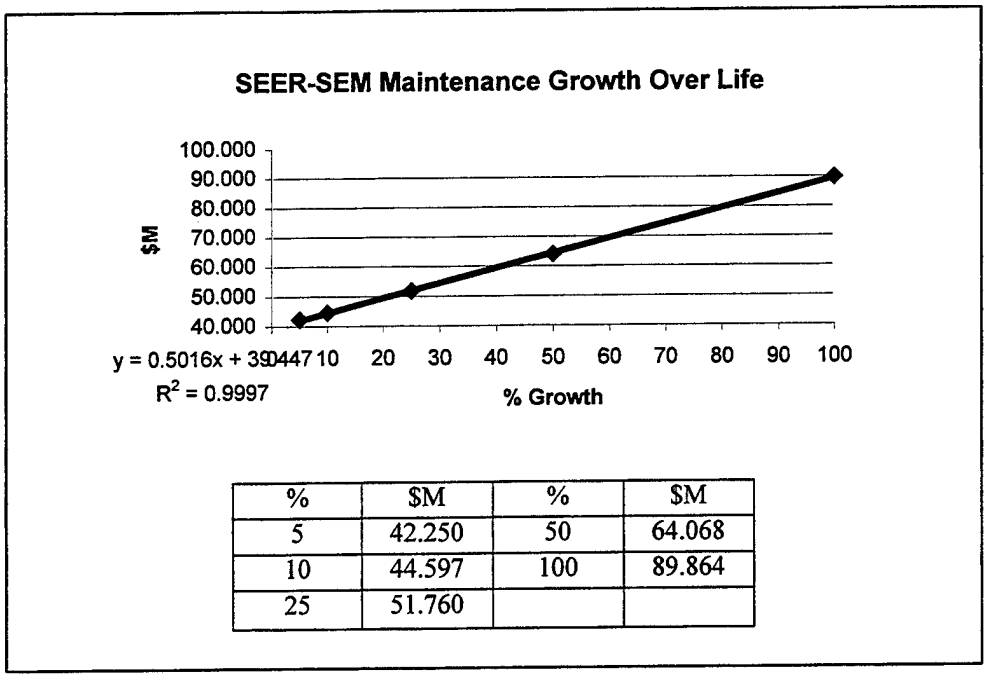
Maintain Total System. Determines whether total size or effective size should be used to estimate maintenance.

(Please see Table 13/Figures 19 - 22 for the percentage changes of a one increment change (decrease and increase) in any maintenance unique inputs. Table 14 illustrates total effort by categories.)



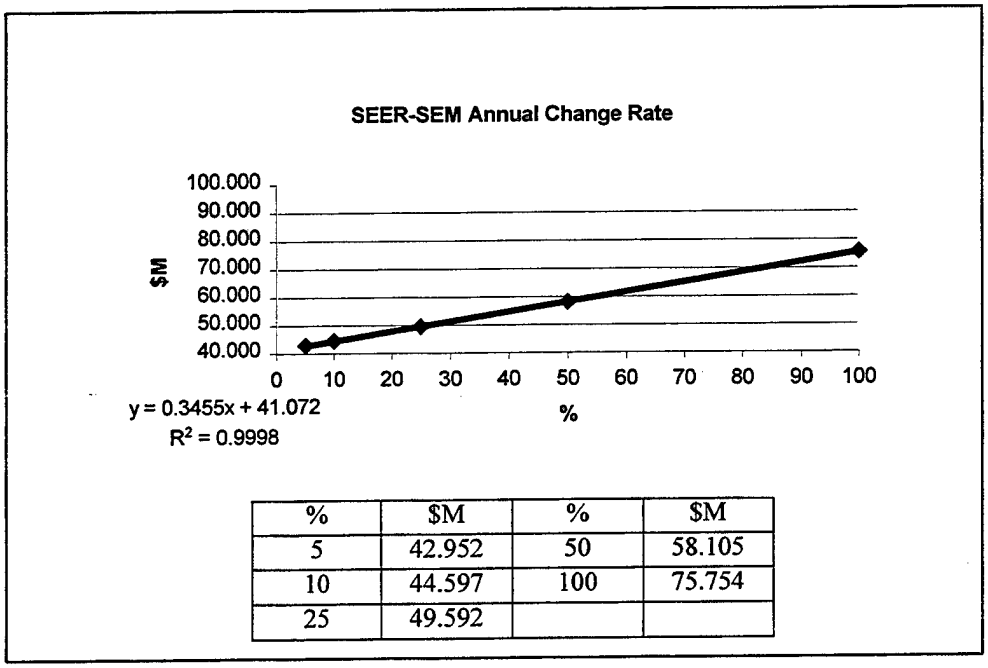
# Sites	$M
1	44.597
2	45.648
5	48.826
10	54.200

Figure 19. SEER-SEM Separate Site Changes



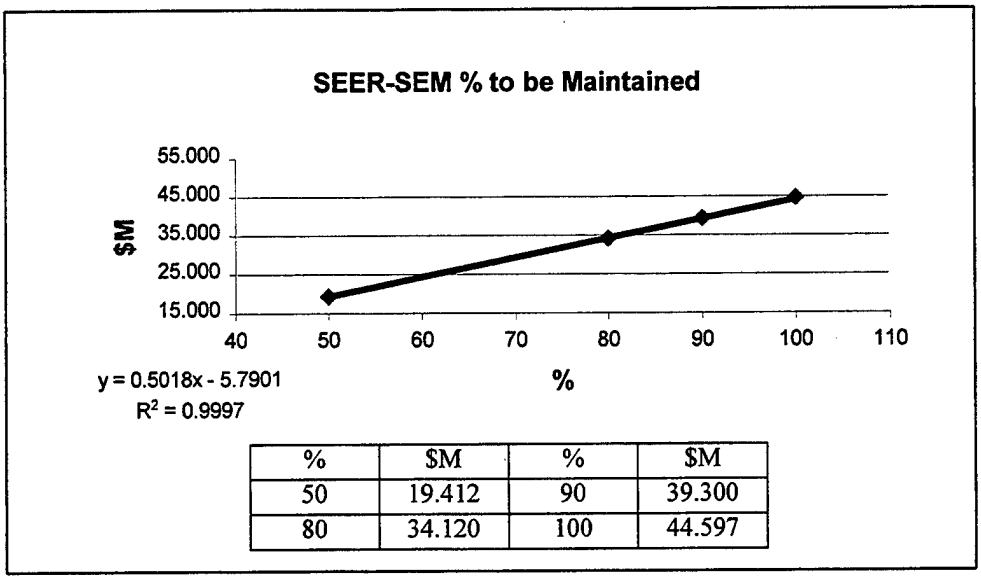
%	$M	%	$M
5	42.250	50	64.068
10	44.597	100	89.864
25	51.760		

Figure 20. SEER-SEM Maintenance Growth Over Life Changes



%	$M	%	$M
5	42.952	50	58.105
10	44.597	100	75.754
25	49.592		

Figure 21. SEER-SEM Annual Change Rate Changes



%	$M	%	$M
50	19.412	90	39.300
80	34.120	100	44.597

Figure 22. SEER-SEM % to be Maintained Changes

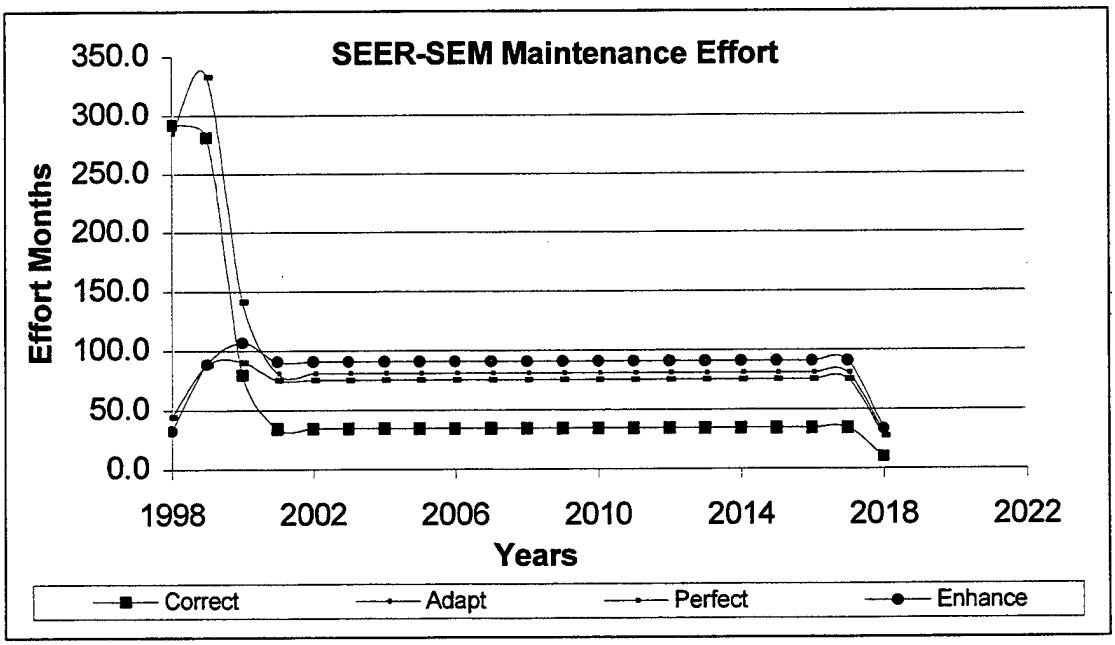
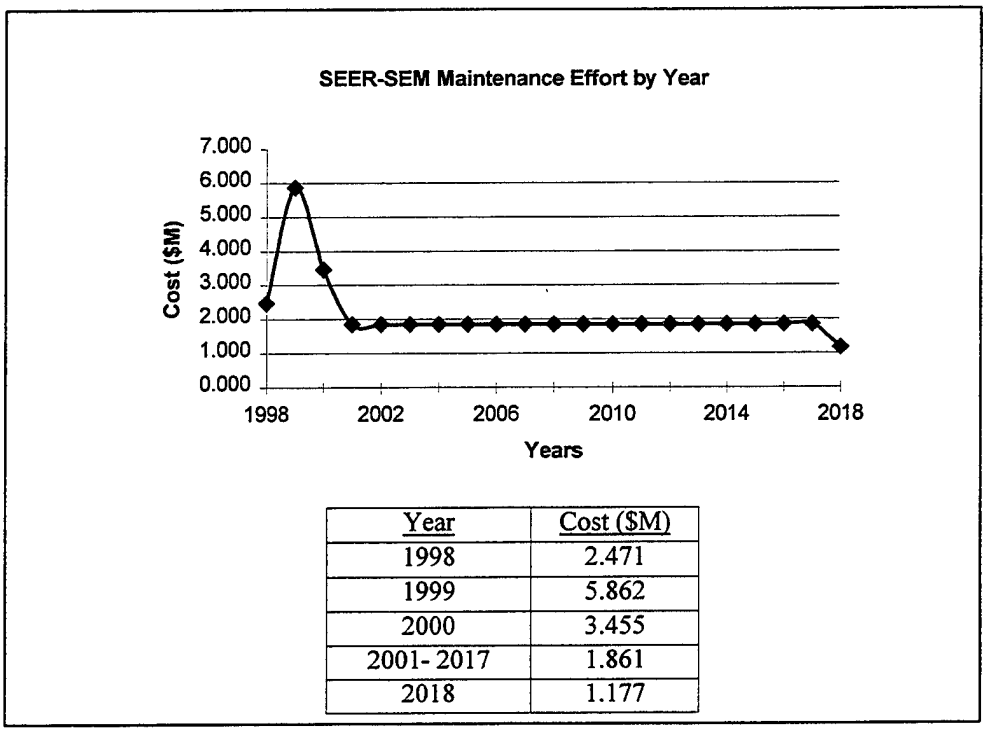


Figure 23. SEER-SEM Maintenance Effort by Category



Year	Cost ($M)
1998	2.471
1999	5.862
2000	3.455
2001- 2017	1.861
2018	1.177

Figure 24. SEER-SEM Maintenance Effort by Year

Table 14. SEER-SEM Total Support Effort by Categories

Fiscal	Effort Months					Base Year	Base Year
Year	Correct	Adapt	Perfect	Enhance	Total	Cost	Cumulafive
1998	292.4	44.2	285.0	32.6	654.2	9,944,596	9,944,596
1999	281.5	88.2	333.2	88.9	791.7	12,034,101	21,978,697
2000	79.7	90.8	141.6	107.1	419.2	6,371,501	28,350,198
2001	34.6	75.4	81.0	91.1	282.2	4,289,502	32,639,700
2002	34.6	75.4	81.0	91.1	282.2	4,289,502	36,929,202
2003	34.6	75.4	81.0	91.1	282.2	4,289,502	41,218,704
2004	34.6	75.4	81.0	91.1	282.2	4,289,502	45,508,206
2005	34.6	75.4	81.0	91.1	282.2	4,289,502	49,797,708
2006	34.6	75.4	81.0	91.1	282.2	4,289,502	54,087,210
2007	34.6	75.4	81.0	91.1	282.2	4,289,502	58,376,712
2008	34.6	75.4	81.0	91.1	282.2	4,289,502	62,666,214
2009	34.6	75.4	81.0	91.1	282.2	4,289,502	66,955,716
2010	34.6	75.4	81.0	91.1	282.2	4,289,502	71,245,218
2011	34.6	75.4	81.0	91.1	282.2	4,289,502	75,534,720
2012	34.6	75.4	81.0	91.1	282.2	4,289,502	79,824,222
2013	34.6	75.4	81.0	91.1	282.2	4,289,502	84,113,724
2014	34.6	75.4	81.0	91.1	282.2	4,289,502	88,403,226
2015	34.6	75.4	81.0	91.1	282.2	4,289,502	92,692,728
2016	34.6	75.4	81.0	91.1	282.2	4,289,502	96,982,230
2017	34.6	75.4	81.0	91.1	282.2	4,289,502	101,271,732
2018	10.2	27.7	29.8	33.4	101.1	1,537,067	102,808,799

SOFTCOST-OO

SoftCost-OO estimates staffing and cost. The specific cost elements are Total Staff, Maintenance Staff, and Sustaining Staff. Cost is effort times the specified hourly pay rate. The model also provides a Present Value estimate based on the estimated cost and user supplied Cost of Money percentage. Total Staff refers to the total number of maintenance and sustaining staff needed for each year. Maintenance Staff is the number of staff needed to perform scheduled changes or upgrades to the system each year. Sustaining Staff is the number of support personnel required to sustain the system each year and is based on the Sustaining Engineering Factor.

No input or output is provided for the cost of updating documentation or for program management. Source Lines of Code (SLOC) or Function Points (FPs) may be used. SLOC excludes comments, but includes data declarations, instantiations, and program specifications. A SoftCost-OO source line is defined by a terminal semicolon with instantiated code counted once. Function Points (FP) must be a value between 25 and 40,000 and represents the size of the system based upon counts of external inputs, external outputs, internal files, operating modes, rendezvous, stimulus/response relationships, external inquires, and external interfaces. The FP counts may be derived from the system or the software requirements specification.

The model comes loaded with three calibration files for using the Ada, Object Oriented, or C++ programming languages. The user may also make a new calibration file if required. The researchers created a new calibration file by copying the scada.cal file into notepad and changing the hours per month to 152 hours, and then saving the file back into SoftCost-OO as thscada.cal. The researchers discovered the model was going

back to the calibration file for every calculation and was not holding changed inputs for the hours per month. Once the calibration file was updated, the hours-per-month was accurate for the research effort.

The model's development inputs may have a significant impact on the support cost. Numerous development inputs changed support cost by more than 15-30% for a one increment change in the factor. See table 15 for development parameter changes.

Software Support unique inputs are:

Nominal Effort (person-months)
Nominal Duration (months)
Sustaining Engineering Factor (%)
Annual Change Traffic (%)
Product Life (years)
Cost of Money (%)

These are defined as follows:

Nominal Effort (person-months). The amount of effort to develop the software system. May use estimated value from SoftCost-OO, actuals from a project, of users estimate.

Nominal Duration (months). The amount of time to develop the software system. May use estimated value from SoftCost-OO, actuals from a project, of users estimate.

Sustaining Engineering Factor (%). An estimate of the percentage of the total O&S effort that will be devoted to sustaining engineering tasks; the amount of sustaining engineering the system will receive during O&S, including user support and training, planned upgrades, and unscheduled repairs. SoftCost-OO defined the Sustaining Engineering Factor as the fraction of the Development workforce that will be assigned to the sustaining engineering task during O&S.

Annual Change Traffic (%). An estimate of the percentage of source code that will undergo change during a typical year.

Product Life (years). The expected life of the software system, 3 to 20 years inclusive.

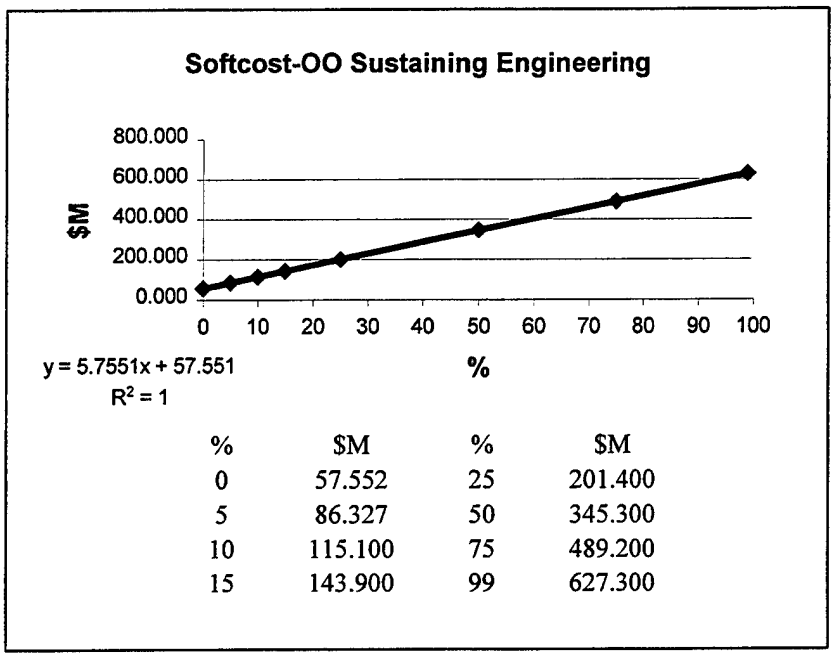
Cost of Money (%). Represents the cost of capital financing. Used for the Present Value calculations.

Table 15. SoftCost-OO Development Input Parameter Changes

SoftCost-OO Development Inputs						
Baseline $57,551,500						
	Chg	Decrease	%Chg	Chg	Increase	%Chg
PROJECT FACTORS						
Type of Software	2 - 2	N/A		2 - 2		
System Architecture	1 - 1	57,551,500	0.0%	1 - 2	74,818,800	30.0%
Number of Organizations Involved	1 - 1	57,551,500	0.0%	1 - 2	60,748,300	5.6%
Organizational Interface Complexity	3 - 2	56,272,800	-2.2%	3 - 4	60,748,300	5.6%
Staff Resource Availability	3 - 2	65,204,900	13.3%	3 - 4	52,990,100	-7.9%
Computer Resource Availability	3 - 2	66,471,000	15.5%	3 - 4	49,630,800	-13.8%
Security Requirements	3 - 2	54,674,400	-5.0%	3 - 4	63,305,700	10.0%
PROCESS FACTORS						
Degree of Standardization	4 - 3	48,364,200	-16.0%	4 - 5	62,388,400	8.4%
Scope of Support	3 - 2	54,674,000	-5.0%	3 - 4	74,818,800	30.0%
Use of Modern Software Methods	3 - 2	72,194,400	25.4%	3 - 4	46,592,700	-19.0%
Use of Peer Reviews	3 - 2	63,305,700	10.0%	3 - 4	54,674,400	-5.0%
Use of Software Tools/Environment	3 - 2	69,003,200	19.9%	3 - 4	53,582,400	-6.9%
Software Tool/Environment Stability	3 - 2	64,459,300	12.0%	3 - 4	55,823,400	-3.0%
PRODUCT FACTORS						
Technology Usage Factor	3 - 2	75,439,200	31.1%	3 - 4	68,972,100	19.8%
Product Complexity	3 - 2	48,920,100	-15.0%	3 - 4	67,910,900	18.0%
Requirements Volatility	3 - 2	50,643,600	-12.0%	3 - 4	66,761,900	16.0%
Degree of Optimization	3 - 2	49,494,600	-14.0%	3 - 4	62,156,700	8.0%
Degree of Real-Time	3 - 2	46,617,500	-19.0%	3 - 4	68,326,900	18.7%
Re-use Benefits	3 - 2	70,260,700	22.1%	3 - 4	46,099,600	-19.9%
Re-use Costs	3 - 2	49,747,300	-13.6%	3 - 4	55,036,400	-4.4%
Database Size	3 - 2	54,099,900	-6.0%	3 - 4	61,003,100	6.0%
PERSONNEL FACTORS						
Number of 00 Projects Completed	1 - 1	57,551,500	0.0%	1 - 1	44,453,900	-22.8%
Analyst Capability	3 - 2	71,941,700	25.0%	3 - 4	49,494,600	-14.0%
Applications Experience	3 - 2	67,910,900	18.0%	3 - 4	51,797,300	-10.0%
Environment Experience	3 - 2	63,305,700	10.0%	3 - 4	48,920,100	-15.0%
Language Experience	3 - 2	67,336,400	17.0%	3 - 4	51,797,300	-10.0%
Methodology Experience	3 - 2	71,362,500	24.0%	3 - 4	49,494,600	-14.0%
Team Capability	3 - 2	63,305,700	10.0%	3 - 4	51,797,300	-10.0%

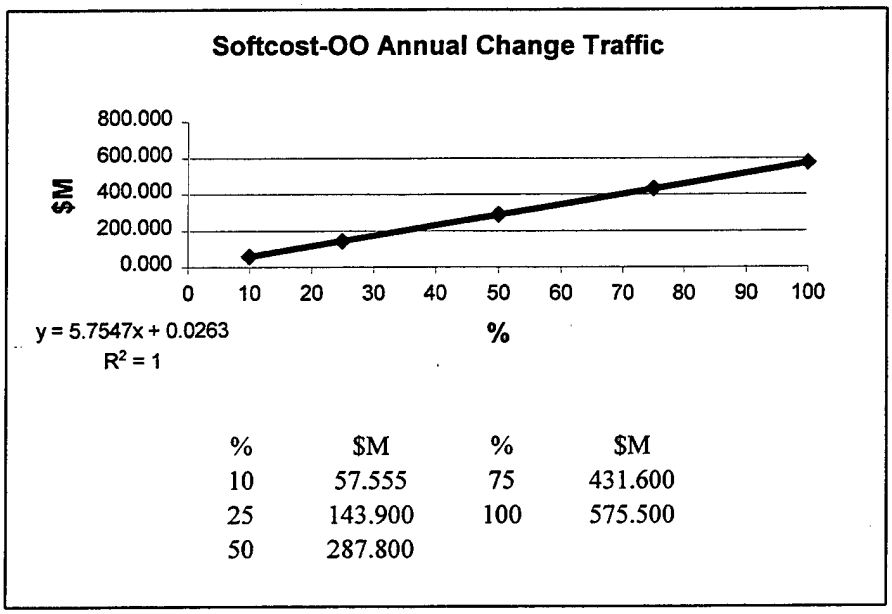
Please see Figures 25 and 26 for the percentage changes or equations. Nominal Effort, nominal duration, and cost of money was not adjusted for this research effort. Figure 27

shows SoftCost-OO's 20-year support cost and Table 16 shows the 20 year support effort.



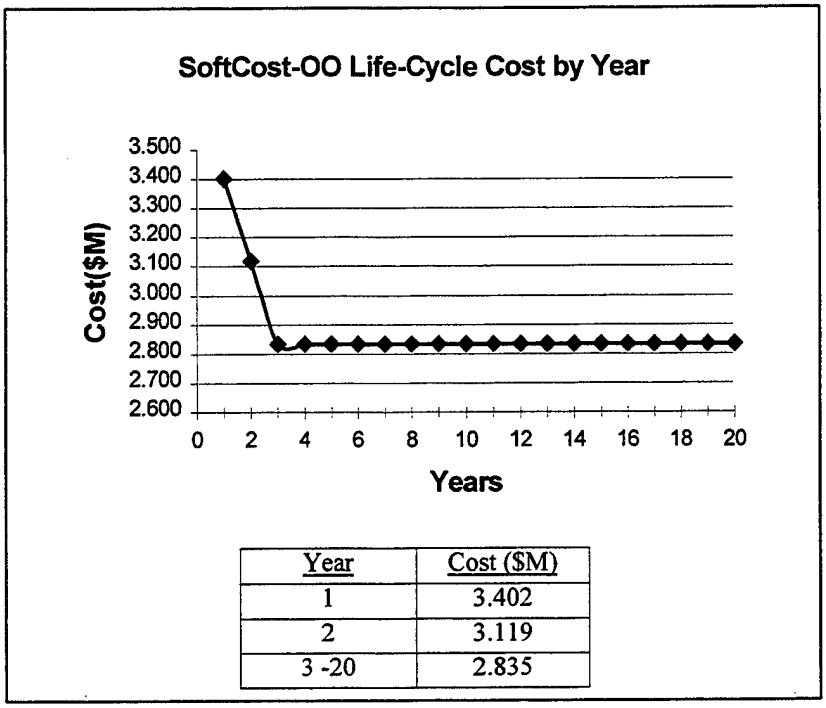
%	$M	%	$M
0	57.552	25	201.400
5	86.327	50	345.300
10	115.100	75	489.200
15	143.900	99	627.300

Figure 25. SoftCost-OO Sustaining Engineering



%	$M	%	$M
10	57.555	75	431.600
25	143.900	100	575.500
50	287.800		

Figure 26. SoftCost-OO Annual Change Traffic



Year	Cost ($M)
1	3.402
2	3.119
3 -20	2.835

Figure 27. SoftCost-OO 20 Year Support Cost

Table 16. SoftCost-OO 20-year Support Effort

Year	Total Staff	Maintenance Staff	Sustaining Staff	Total Cost
1	18.7	18.7	0.0	3,402,100.0
2	17.1	17.1	0.0	3,118,600.0
3	15.5	15.5	0.0	2,835,000.0
4	15.5	15.5	0.0	2,835,000.0
5	15.5	15.5	0.0	2,835,000.0
6	15.5	15.5	0.0	2,835,000.0
7	15.5	15.5	0.0	2,835,000.0
8	15.5	15.5	0.0	2,835,000.0
9	15.5	15.5	0.0	2,835,000.0
10	15.5	15.5	0.0	2,835,000.0
11	15.5	15.5	0.0	2,835,000.0
12	15.5	15.5	0.0	2,835,000.0
13	15.5	15.5	0.0	2,835,000.0
14	15.5	15.5	0.0	2,835,000.0
15	15.5	15.5	0.0	2,835,000.0
16	15.5	15.5	0.0	2,835,000.0
17	15.5	15.5	0.0	2,835,000.0
18	15.5	15.5	0.0	2,835,000.0
19	15.5	15.5	0.0	2,835,000.0
20	15.5	15.5	0.0	2,835,000.0

SoftCost-OO's Life Cycle Cost Model, SoftCost-LC, uses the following equation when performing an unconstrained estimate:

$$\mathbf{MM_{an} = A * ((MM_{nom} * EAF_m * ACT) + (MM_{nom} * SEF_a))}$$

Where:	MM_{an}	= Annual O&S Phase Effort
	A	= Calibration Coefficient
	MM_{nom}	= Nominal Development Effort
	EAF_m	= Effort Adjustment Factor
	ACT	= Annual Change Traffic
	SEF_a	= Sustaining Engineering Factor

$$\mathbf{MM_{nom} = TC * (SIZE)^{P1}}$$

Where:	TC	= Effort Technology Constant
	SIZE	= Size of Software System
	P1	= Effort Exponent

The acronyms are defined as follows:

Calibration Coefficient. The combined effects of all the SoftCost-OO constants, cost drivers, and multipliers. Estimated by SoftCost-OO, but may be manually input.

Nominal Development Effort. Amount of effort required to initially develop the software system. Estimated by SoftCost-OO, but may be manually input.

Effort Adjustment Factor. Determined by SoftCost by recalibrating seven parameters (usage factor, degree of real-time, modern programming methods, computer resource availability, staff resource availability, degree of reuse, and use of software tools/environment).

Annual Change Traffic. An estimate of the percentage of source code that will undergo change during a typical year.

Sustaining Engineering Factor. An estimate of the percentage of the total O&S effort that will be devoted to sustaining engineering tasks.

During load balancing estimates, the equation becomes constrained.

$A * (MM_{nom} * SEF_a) = C\text{-}A * (MM_{nom} * EAF_m * ACT)$ and

$A * (MM_{nom} * EAF_m * ACT) = C\text{-}A * (MM_{nom} * SEF_a)$

Where C = a constant representing the workforce level.

SoftEst.

The SoftEst model gave an estimate of $91.2M for the 20-year maintenance period. The support costs were not categorized or broken out into the maintenance, enhancement, and growth category in the maintenance report; however, SoftEst assumes that the maintenance effort is comprised of both error corrections and small software enhancements which are new. The ACT method used by SoftEst to calculate the support cost/effort is comprised of the following equation:

Mmam = (MMnom) ACT II (MFi).

MMnom = equal to A * ((KDSI) ^ B) * Fi).

A = coefficient

B = exponent

MFi = a product of a group of maintenance environmental factors

Fi = a product of a group of development environmental factors

KDSI = the delivered source instructions in thousands

ACT = Annual Change Traffic

The coefficient A, the exponent B, and the MFi value are determined by statistical means from a group of previously completed projects. Annual Change Traffic (ACT) represents the amount source instructions that will be changed on a yearly basis in the support effort. The change in source instructions can be comprised of both modifications and additions to the original software. MFi is a set of maintenance environment factors that are similar to the standard environmental factors with the exception to modern programming practices (MOPD) and required reliability (RELY). These two parameters have different maintenance look-up settings/values assigned to them for estimating the support costs/effort. The underlying logic for the usage of the two development parameters

for the support effort calculation is that modern programming practices and required reliability will normally result in less initial errors in the developed effort of the software, which in turn leads to less support in the future. A support cost value is then given for each year that support is to be performed for the required number of years that the user desires to support the given system. The model does not include a line item break out showing the costs associated with documentation or the cost of program management, thus a lump sum cost for the entire effort is shown for the support. It is assumed in the model that required reliability is highly correlated with the level and type of documentation required with a given project. A high reliability level would normally result in a formal level of reviews and documentation. This could be used to adjust for the varying levels of documentation the system.

The first step of the analysis was to identify the categories of parameters that affect the support effort. SoftEst has two main input sections: environment and maintenance. Softest also has three additional input screens: size, cost, and architecture. The first section analyzed was the environmental parameter section, which represents or characterizes the software project. Table 17 represents the changes (one unit increase/decrease in the parameter values) made to the environment parameters in CSCI 1. The environment parameter that was identified to have the greatest impact on the support effort/cost in increasing the parameter value for CSCI 1 was the required reliability parameter. The required reliability parameter quantifies the extra effort that is often associated with higher levels of reliability in the end product. If the software is going into a system where the loss of life is possible, then extra effort will be expended in the verification and validation activities. This was one of the values that previously

descriptions/values for required reliability is from slight inconvenience (VL) to possible loss of life and trusted computer security systems extra high (XH). An increase in the required reliability parameter value from HI to VH resulted in a cost increase to $6.9M or a 5.66% increase from the baseline cost. A decrease in the required reliability parameter value from HI to NM resulted in a cost decrease to $6.3M or a 3.40% decrease from the baseline cost. Figure 28 shows the variations in the required reliability values compared to the support cost for CSCI 1.

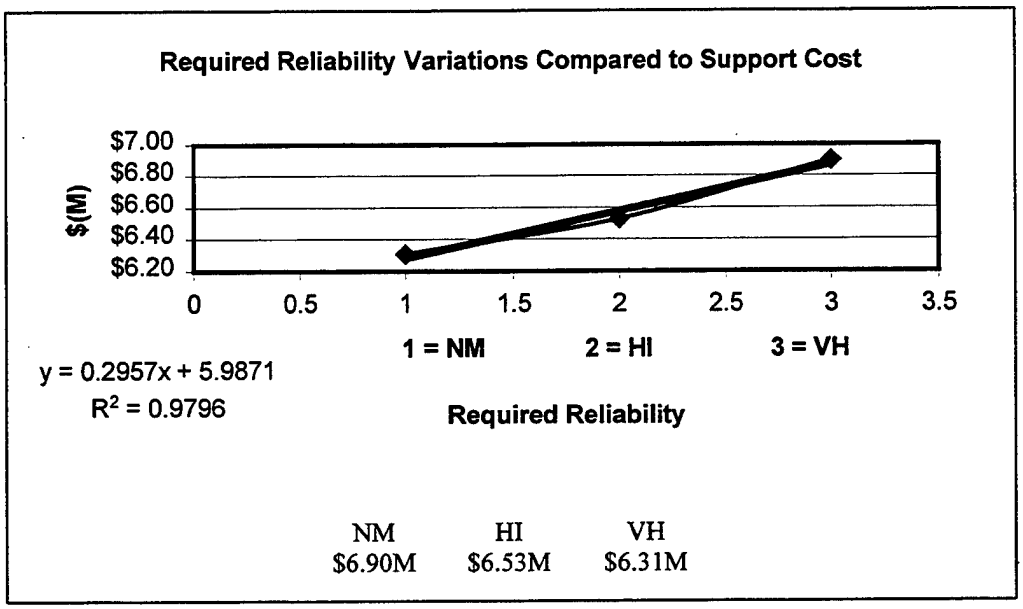


Figure 28. Required Reliability Compared to Support Cost CSCI 1.

The environment parameter that was identified to have the greatest impact on the support effort/cost in decreasing the parameter value for CSCI 1 was the required analyst capability parameter. The analyst capability parameter is a measure of the effectiveness of the system engineering and software design team. The analysts perform a review of the project requirements and specifications, define the architecture, and produce the preliminary design specifications. The range of parameter descriptions for analyst

capability is from new personnel with no demonstrated capability (VL) to strong team with many highly capable personnel (VH). A decrease in the experience attribute value from NM to LO resulted in a cost decrease to $6.9M or a 4.95% increase from the baseline cost. An increase in the analyst capability parameter value from NM to HI resulted in a cost decrease to $6.3M or a 3.65% decrease from the baseline cost. Figure 29 shows the variations in the analyst capability parameter values compared to the support cost for CSCI 1.

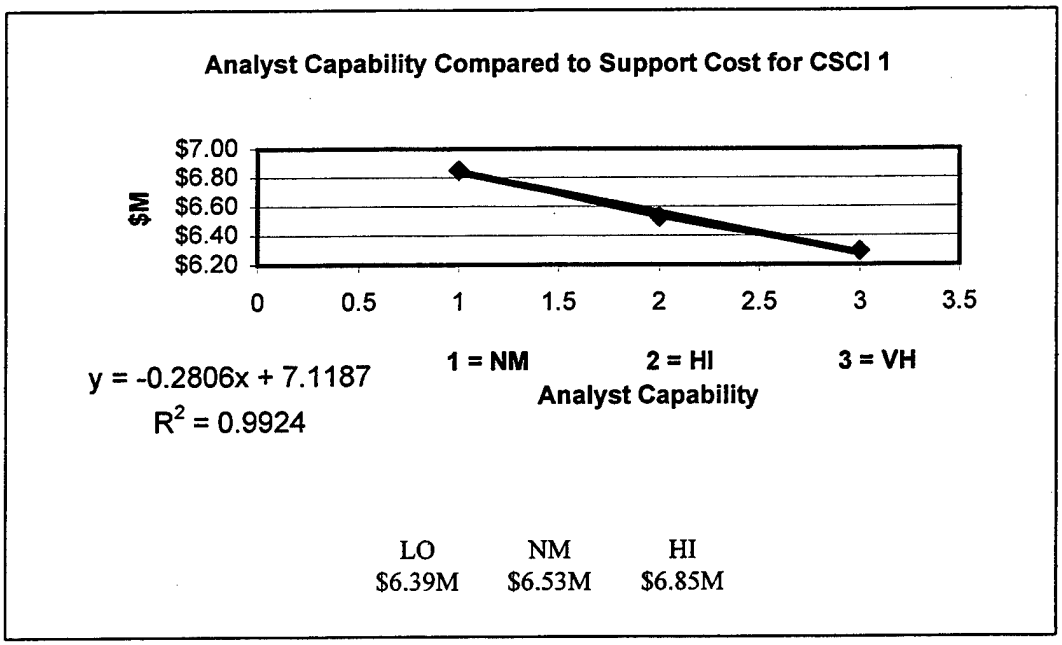


Figure 29. Analyst Capability Compared to Support Cost CSCI 1.

The environment parameter that was identified to have the greatest negative impact on the support effort/cost in decreasing the parameter values for CSCI 1 was the product complexity parameter. The product complexity parameter quantifies the extra effort needed with the complexity of the software product being developed. The range of parameter descriptions/values for product complexity is from offline simple print

routines, small utilities (VL) to signal processing algorithms, complex algorithms, CSCI's greater than approximately 150 KDSI, or applications with high security levels. A decrease in the product complexity parameter value from NM to LO resulted in a cost decrease to $6.3M or a 3.91% decrease from the baseline cost, see table 17. An increase in the product complexity attribute value from NM to HI resulted in a cost increase to $6.8M or a 3.91% increase from the baseline cost. Figure 30 shows the variations in the product complexity values compared to the support cost for CSCI 1.

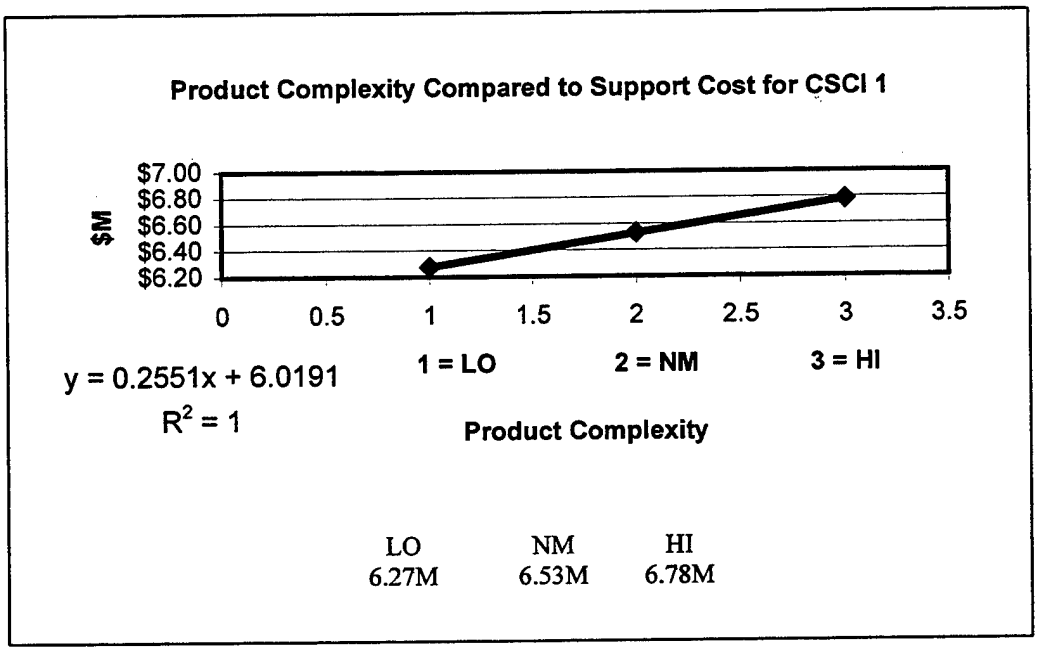


Figure 30. Product Complexity Compared to Support Cost CSCI 1.

In summary for CSCI 1, when the parameters are changed by a one-unit decrease, seven out of the eighteen parameters decreased the support cost, while the remaining attributes increased the support cost. When the parameters were changed by a one-unit decrease, seven out of the nineteen parameters decreased the support cost, while eight attributes increased the support cost and three parameters were not affected.

Table 17. SoftEst Parameter Changes and Effects on Support Costs CSCI 1

	CSCI 1 Parameters		Year 1 Support Cost $6,529,209				
	Parameter	Parameter Change Dec.	New Cost	Change	Parameter Change Inc.	New Cost	Change
1	Analyst Capability	NM - LO	$6,852,285	4.95%	NM - HI	$6,291,154	-3.65%
2	Programmer Capability	NM - LO	$6,818,276	4.43%	NM - HI	$6,291,154	-3.65%
3	Application Experience	NM - LO	$6,750,260	3.39%	NM - HI	$6,376,172	-2.34%
4	Virtual Machine Experience	NM - LO	$6,699,249	2.60%	NM - HI	$6,359,170	-2.60%
5	Language Experience	NM - LO	$6,648,238	1.82%	NM - HI	$6,444,188	-1.30%
6	Processing Time Constraints	NM - LO	$6,529,209	0.00%	NM - HI	$6,716,253	2.86%
7	Hardware Storage Constraints	NM - LO	$6,529,209	0.00%	NM - HI	$6,631,234	1.56%
8	Virtual Machine Volatility	NM - LO	$6,308,156	-3.39%	NM - HI	$6,784,268	3.91%
9	Development Turn Around Time	NM - LO	$6,308,156	-3.39%	NM - HI	$6,648,238	1.82%
10	Requirements Volatility	NM - LO	$6,376,172	-2.34%	NM - HI	$6,852,285	4.95%
11	Required Reliability	HI - NM	$6,307,418	-3.40%	HI – NM	$6,898,860	5.66%
12	Data Base Size	NM - LO	$6,427,816	-1.55%	NM - HI	$6,665,241	2.08%
13	Product Complexity	NM - LO	$6,274,149	-3.91%	NM - HI	$6,784,268	3.91%
14	Design For Reuse	NM - LO	$6,529,209	0.00%	NM - HI	$6,699,249	2.60%
15	Modern Development Practices	NM - LO	$6,699,249	2.60%	NM - HI	$6,376,172	-2.34%
16	Use Of Automated Tools	NM - LO	$6,699,249	2.60%	NM - HI	$6,376,172	-2.34%
17	Classified Environment	UNCL - CL	$6,699,249	2.60%	UNCL – CL	$6,699,249	2.60%
18	Schedule Constraints	NM unable to change	-	-	NM unable to change	-	-
19	Platform Risk	MAN AIR – UNMAN AIR	$6,340,276	-2.89%	MAN AIR – UNMAN SP	$6,718,142	2.89%

The environment parameter that was identified to have the greatest impact on the support effort/cost in increasing the parameter value for CSCI 2 was once again the required reliability parameter. An increase in the required reliability parameter value from HI to VH resulted in a cost increase to $7.2M or a 10.95% increase from the baseline cost. A decrease in the required reliability parameter value from HI to NM resulted in a cost decrease to $6.1M or a 6.57% decrease from the baseline cost. Figure 31 shows the variations in the required reliability values compared to the support cost for CSCI 2.

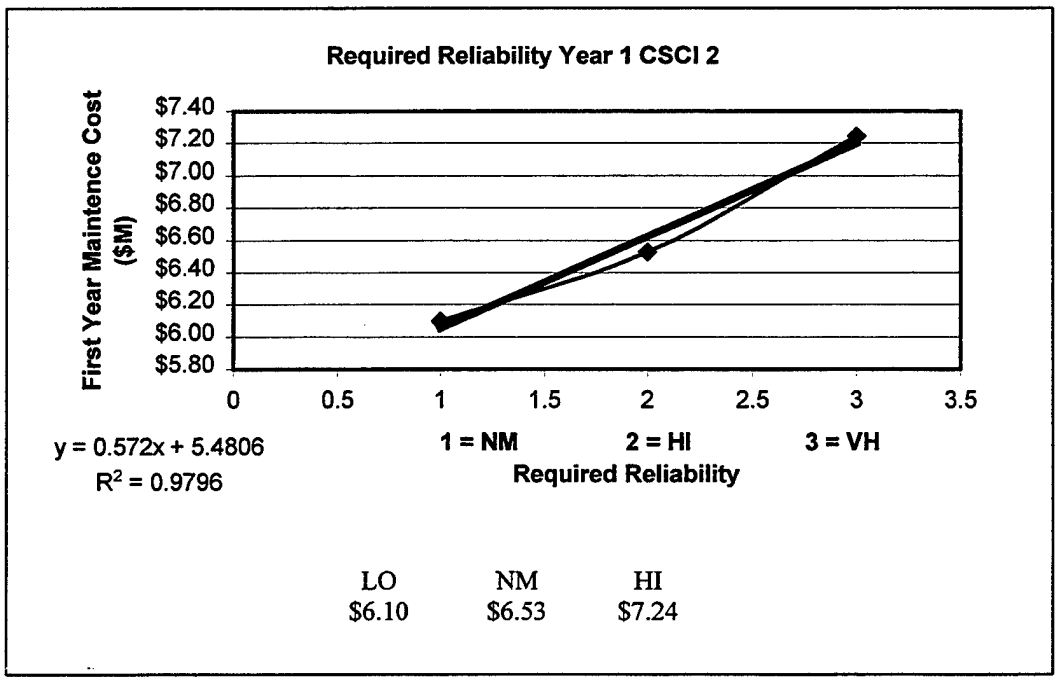


Figure 31. Required Reliability Compared to Support Cost CSCI 2.

The second largest percentage change for an increase in the environment parameters for CSCI 2 was the requirements volatility parameter. The requirements volatility parameter measures the amount of project design and development rework that is the result of changes in the users requirements for system. The changes will normally

start with an engineering change proposal from a contractor or a request for change from the user. For a military program a setting of high should be used as a minimum. An increase in the requirements volatility parameter value from NM to HI resulted in a cost increase to $7.2M or a 9.57% increase from the baseline cost. A decrease in the required requirements volatility parameter value from NM to LO resulted in a cost decrease to $6.2M or a 4.53% decrease from the baseline cost. Figure 32 shows the variations in the requirements volatility parameter values compared to the support cost for CSCI 2.

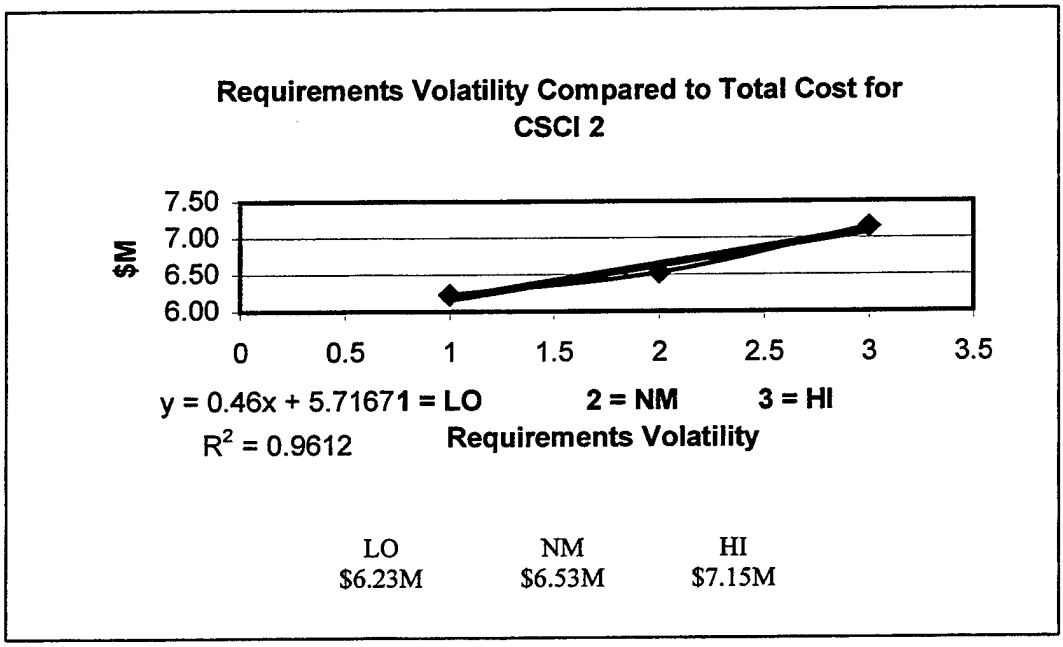


Figure 32. Requirements Volatility Compared to Support Cost CSCI 2.

The environment parameter that was identified to have the greatest impact on the support effort/cost in decreasing the parameter value for CSCI 2 was the analyst capability parameter. A decrease in the analyst capability parameter value from NM to LO resulted in a cost increase to $7.2M or a 9.57% increase from the baseline cost. An increase in the required analyst capability parameter value from NM to HI resulted in a

cost decrease to $6.1M or a 7.05% decrease from the baseline cost. Figure 33 shows the variations in the analyst capability values compared to the support cost for CSCI 2.

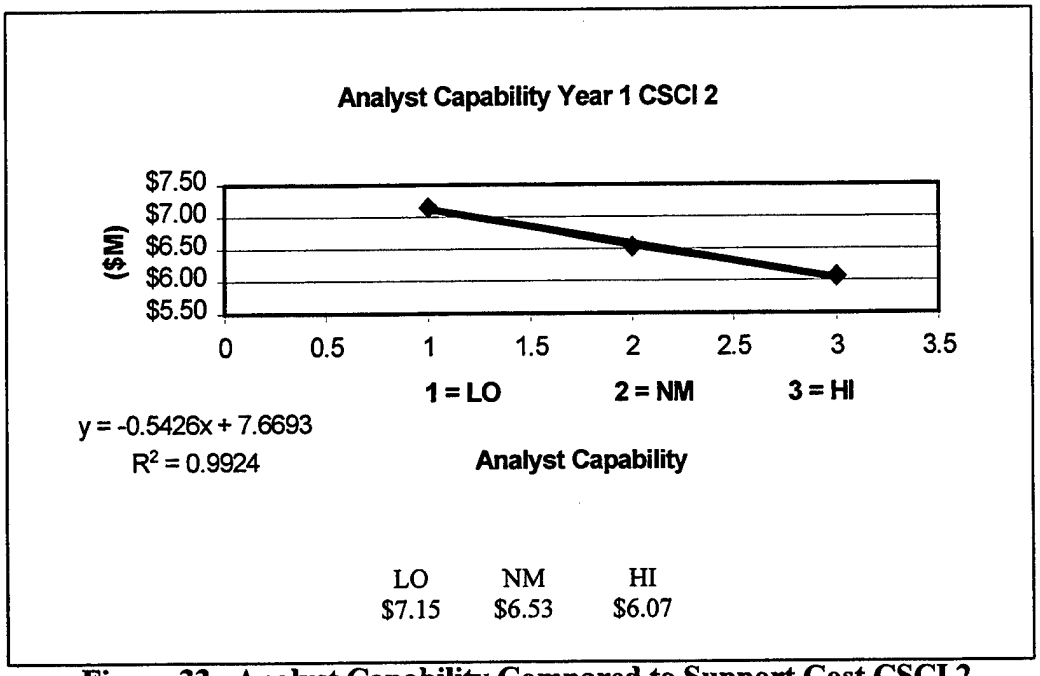


Figure 33. Analyst Capability Compared to Support Cost CSCI 2.

The second largest percentage change for a decrease in the environment parameter that was identified to have an impact on the support effort/cost in decreasing the parameter value for CSCI 2 was the programmer capability parameter. The programmer capability parameter is a measure of the effectiveness of the software development team. A decrease in the programmer capability parameter value from HI to NM resulted in a cost increase to $7.1M or an 8.20% increase from the baseline cost. An increase in the required programmer capability parameter value from HI to VH resulted in a cost decrease to $5.9M or a 9.37% decrease from the baseline cost. Figure 34 shows the variations in the programmer capability values compared to the support cost for CSCI 2.

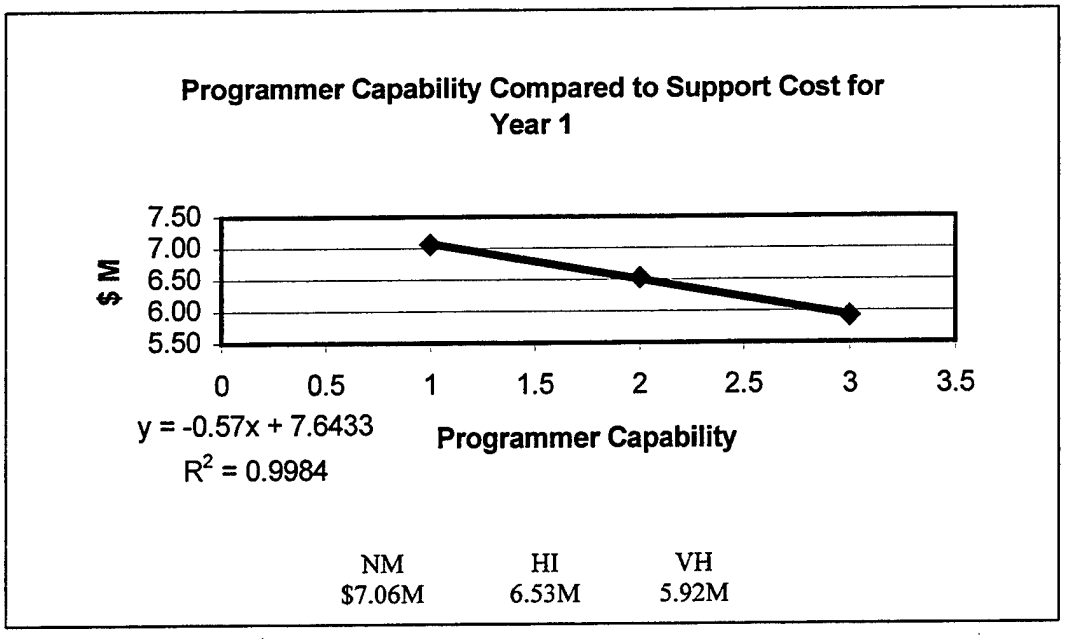


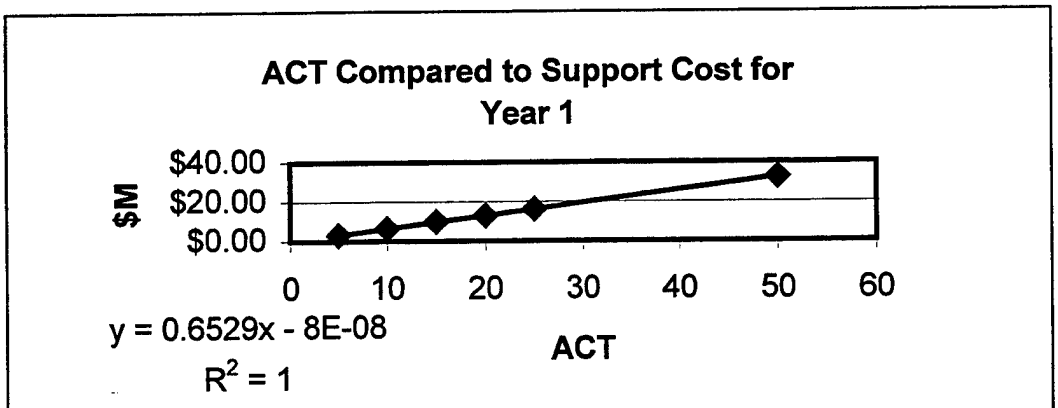
Figure 34. Programmer Capability Compared to Support Cost CSCI 2.

In summary for CSCI 2, when the parameters were changed by a one-unit decrease, seven out of the nineteen adjustable parameters decreased the support cost, with eight attributes increasing the support cost and three parameters not affecting the support cost. When the parameters were changed by a one-unit increase, six out of the eighteen parameters decreased the support cost, while eleven attributes increased the support cost, and one parameter did not affect the cost.

The second input area analyzed was the maintenance section. The first two parameters in the maintenance area did not influence the support cost: software understanding and software assimilation. Thus, leading the researchers to believe that the parameters were not functioning properly in the model. The third parameter, number of years for the support effort, appeared to be functioning but the estimate values were the same for each year irrespective of the number of years of maintenance. Thus, the first

five years of maintenance for a ten-year maintenance period were equal to a five-year maintenance period.

The fourth parameter analyzed was annual change traffic. Annual Change Traffic (ACT) represents the amount source instructions that will be changed on a yearly being comprised of both modifications and additions to the original software. The annual change traffic parameter was changed from 10 to 5, which resulted in a 50% decrease in the first year of support costs used as a baseline. When the value was increased from 10 to 15 a 50% increase in the first year of support cost was identified. The ACT parameter was then increased to 15, 25, and 50. The results were a 100%, 150%, and 400% increase in the first year support costs respectively. Figure 35 shows the variations in the ACT values compared to the support cost for CSCI 2. Table 18 the parameter changes and the effects on CSCI 2.



ACT	5	10	15	20	25	50
First Year Cost	$3.26M	$6.53M	$9.79M	$13,06M	$16,32M	$32.65M
% Change	-50.00%	0.00%	50.00%	100.00%	150.00%	400.00%

Figure 35. Annual Change Traffic Compared to Support Cost CSCI 2.

Table 18. SoftEst Parameter Changes and Effects on Support Costs CSCI 2

	CSCI 2 Parameters	Year 1 Support Cost $6,529,209					
	Parameter	Parameter Change Dec.	New Cost	Change	Parameter Change Inc.	New Cost	Change
1	Analyst Capability	NM - LO	$7,154,070	9.57%	NM - HI	$6,068,784	-7.05%
2	Programmer Capability	HI - NM	$7,064,586	8.20%	HI - VH	$5,917,348	-9.37%
3	Application Experience	NM - LO	$6,956,746	6.55%	NM - HI	$6,233,222	-4.53%
4	Virtual Machine Experience	NM - LO	$6,858,084	5.04%	NM - HI	$6,200,334	-5.04%
5	Language Experience	HI - NM	$6,702,300	2.65%	HI - VH	$6,529,209	0.00%
6	Processing Time Constraints	NM - LO	$6,529,209	0.00%	NM - HI	$6,890,972	5.54%
7	Hardware Storage Constraints	NM - LO	$6,529,209	0.00%	NM - HI	$6,726,533	3.02%
8	Virtual Machine Volatility	NM - LO	$6,101,672	-6.55%	NM - HI	$7,022,521	7.56%
9	Development Turn Around Time	NM - LO	$6,101,672	-6.55%	NM - HI	$6,759,422	3.53%
10	Requirements Volatility	NM - LO	$6,233,222	-4.53%	NM - HI	$7,154,070	9.57%
11	Required Reliability	HI - NM	$6,100,242	-6.57%	HI - NM	$7,244,154	10.95%
12	Data Base Size	NM - LO	$6,331,884	-3.02%	NM - HI	$6,792,309	4.03%
13	Product Complexity	NM - LO	$6,035,898	-7.56%	NM - HI	$7,022,521	7.56%
14	Design For Reuse	NM - LO	$6,529,209	0.00%	NM - HI	$6,858,084	5.04%
15	Modern Development Practices	NM - LO	$6,858,084	5.04%	NM - HI	$6,233,222	-4.53%
16	Use Of Automated Tools	NM - LO	$6,858,084	5.04%	NM – HI	$6,233,222	-4.53%
17	Classified Environment	UNCL - CL	$6,858,084	5.04%	UNCL – CL	$6,858,084	5.04%
18	Schedule Constraints	NM unable to change	-	-	NM unable to change	-	-
19	Platform Risk	MAN AIR – UNMAN AIR	$6,163,793	-5.60%	MAN AIR – UNMAN SPACE	$6,894,625	5.60%

Figure 36 represents the support costs over the twenty years that the support effort will continue. The first three years of support costs allow for a transition period for the software. This allows for residual errors to be found before a consistent steady state is reached with the software support efforts. The maintenance transition factors used for calculating the delta in the support cost effort for the first three years are 1.5 for the first year, 1.3 for the second year, and 1.15 for the third year.

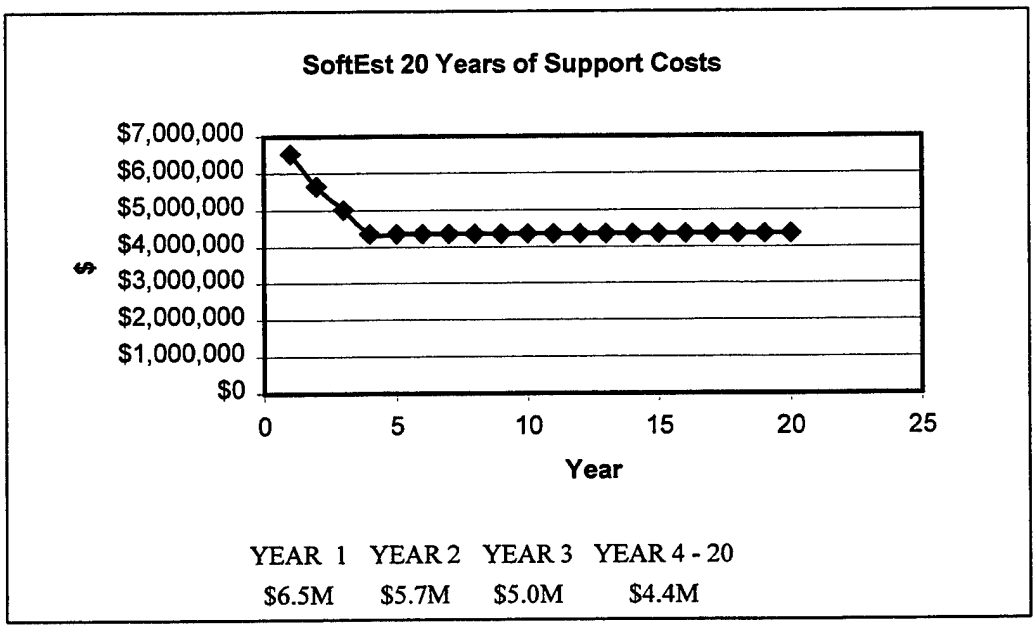


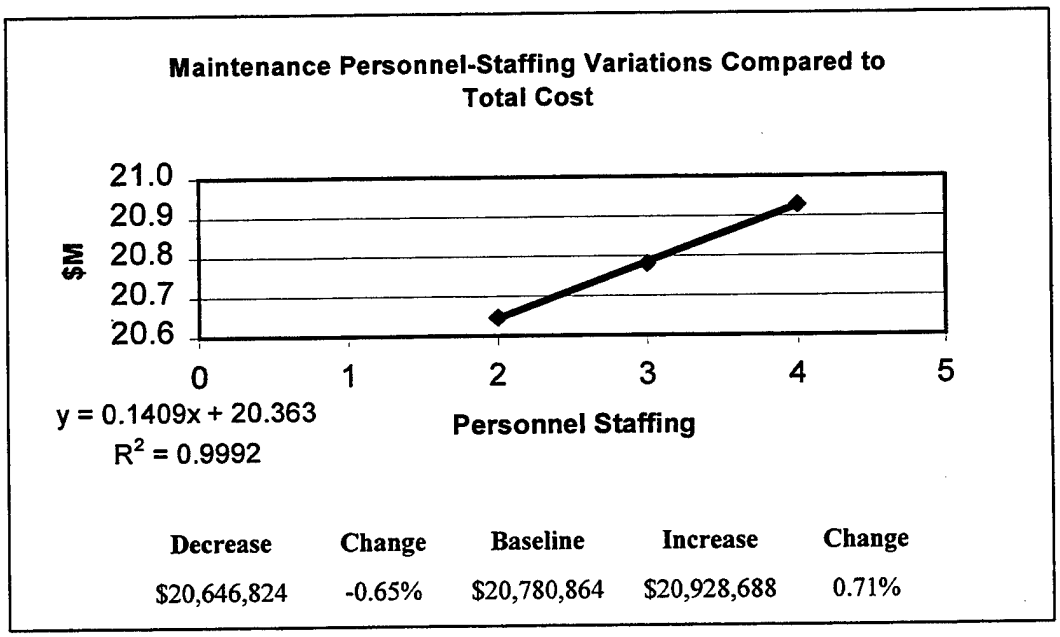
Figure 36. SoftEst 20 Year of Support Costs

KnowledgePLAN 2.0

KnowledgePLAN 2.0 gave an estimate of $20.8M for a 5-year support period with the given base case inputs. The period analyzed in KnowledgePLAN 2.0 was 15 years shorter than the other models due to the 5-year template that the model uses to estimate the support effort/costs. (Software Productivity Research (SPR) is currently in the process of updating KnowledgePLAN 2.0 to handle a 20-year support period through a 20-year template.) The model does separately consider the cost of documentation, however, it is considered in the maintenance documentation reviews of the maintenance manuals. Maintenance management is broken out on a single line item under central maintenance. Maintenance management in KnowledgePLAN 2.0 refers to the normal supervisory tasks associated with managing technical staff, including hiring, appraising, handling budgets, expense tracking, and other associated activities. KnowledgePLAN 2.0 does not directly use SLOC to estimate the support effort/cost, but rather converts the SLOC input into International Function Point Users Group (IFPUG) Function Points. The attribute options in the model are one of the primary sources of adjustments to the estimates calculated by SPR KnowledgePLAN 2.0. The values ranged from 1 (good) to 5 (bad) with a value of 3 being average or nominal. Thus, a response of 3.0 is by convention assumed to have little or no impact on the knowledge base generated estimate. If no value is provided for an attribute, then the edit field shows N/A and the attribute is treated as though it had a value of 3.0. The manner in which the maintenance attributes are defined can have a significant effect on the project quality and productivity as well as time and cost. The maintenance attributes enables the user to characterize the maintenance side of the project. These attributes have a large influence on defect repair

and user support in the maintenance stage of the software.

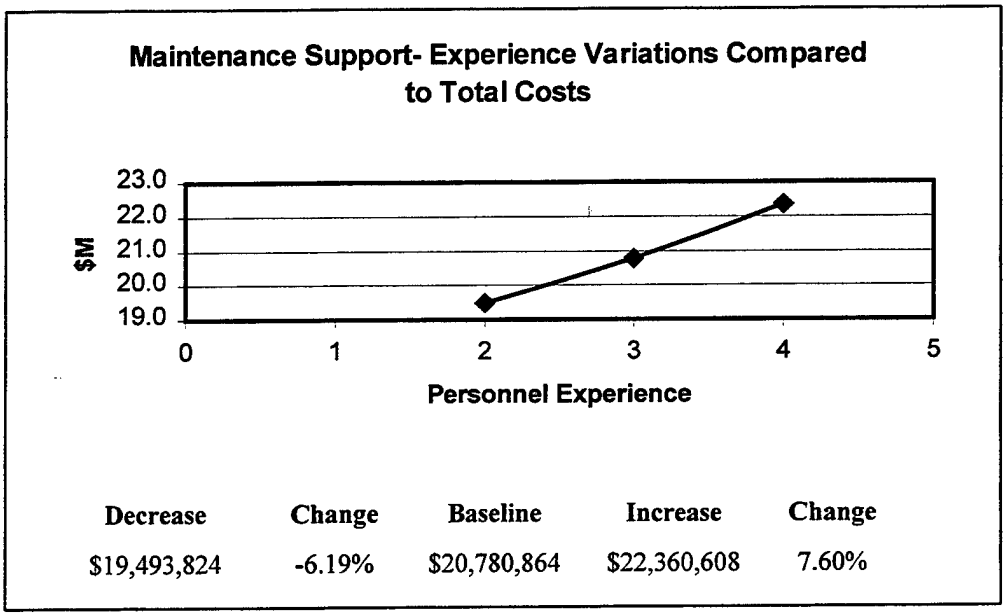
The first maintenance category addressed in the Knowledge Plan 2.0 model was personnel. The attributes for maintenance personnel affect the performance of defect repair and user support in the post production phases of the software development cycle. Experience of personnel is as critical to quality and productivity in maintenance as it is in development effort of the software. The first attribute parameter analyzed within the personnel category was maintenance personnel staffing. The responsibility of bug fixing can vary from project to project. It can be done by a dedicated staff or it can be performed informally by developers. This can have a significant impact on productivity and quality. The range of parameter descriptions/values is from all maintenance being performed by full time professional maintenance personnel (1) to all maintenance being performed by development personnel (5). Figure 37 shows the variations in the staffing values compared to the support costs.



Decrease	Change	Baseline	Increase	Change
$20,646,824	-0.65%	$20,780,864	$20,928,688	0.71%

Figure 37. Maintenance Personnel Staffing Compared to Total Costs.

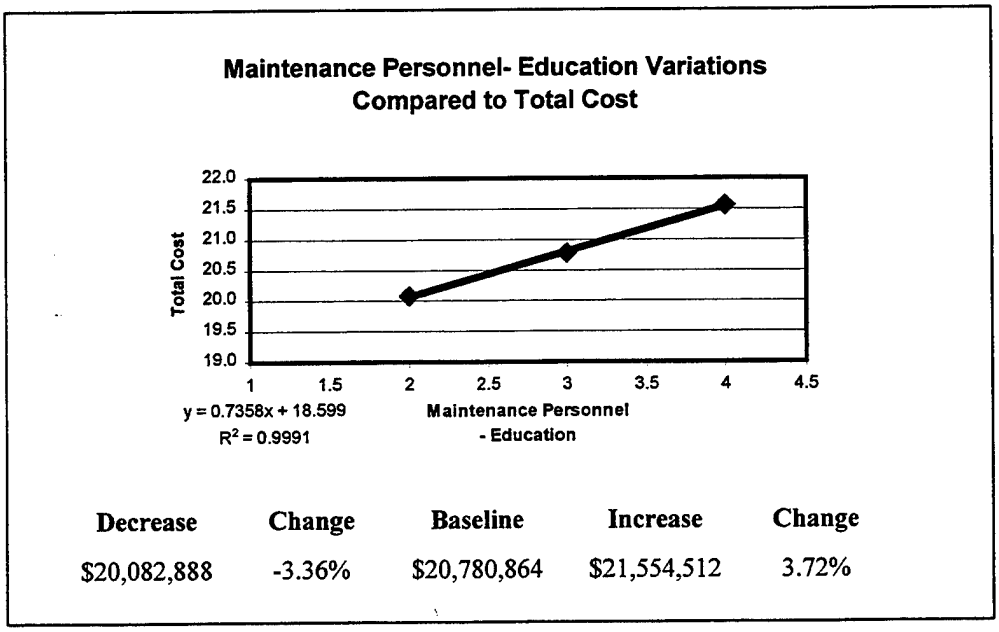
The second attribute analyzed within the personnel category was maintenance personnel experience. Maintenance personnel experience is the amount of experience that an individual has in system that is to be maintained. The experience of maintenance staff in relation to product maintenance can greatly enhance or impair maintenance productivity. The experience parameter was found to have an effect on the support effort/costs by increasing or decreasing the attribute's value. The range of the attribute's descriptions/values is from all personnel being experts in the system to be maintained (1) to all personnel being new to the system being maintained (5). An increase in the experience value from 3.0 to 4.0 resulted in a cost increase to $22.4M or a 7.6% increase from the baseline cost. A decrease in the experience value from 3.0 to 2.0 resulted in a cost decrease to $19.5M or a 6.19% decrease from the baseline cost. Figure 38 shows the variations in the experience values compared to the support costs.



Decrease	Change	Baseline	Increase	Change
$19,493,824	-6.19%	$20,780,864	$22,360,608	7.60%

Figure 38. Personnel Experience Compared to Total Cost

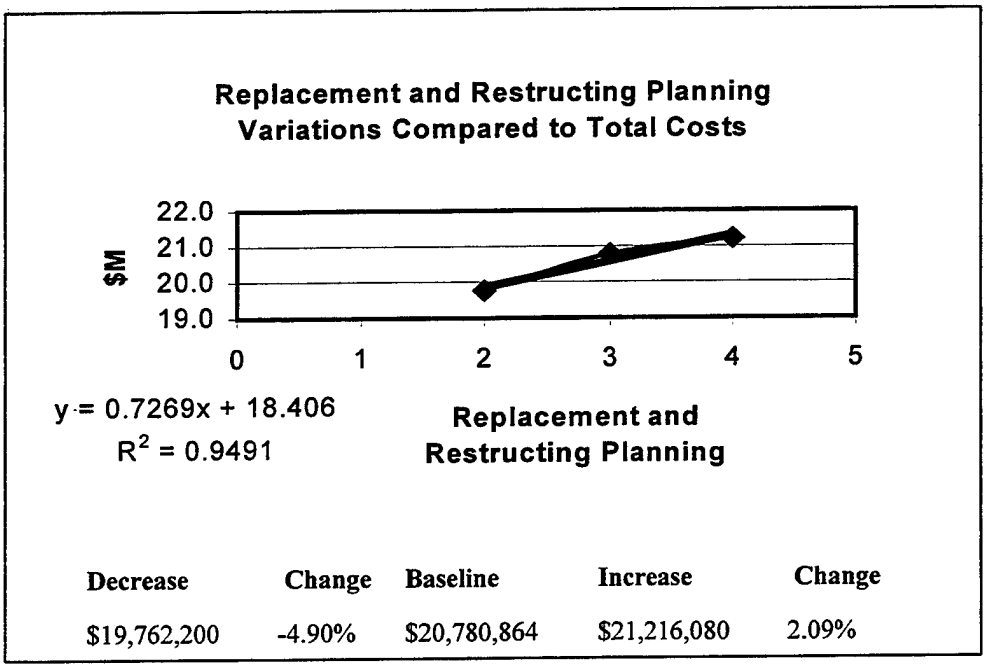
The third attribute analyzed within the personnel category was maintenance personnel education. Maintenance personnel education is the amount of education that an individual has in system that is to be maintained. Training given to the maintenance personnel, prior to the start of maintenance, can affect maintenance productivity. On the job training should not be considered applicable in this context. The experience parameter was found to have an affect on the support effort/costs by increasing or decreasing the attribute's value. The range of the attribute's descriptions/values is from maintenance training is not required for the project (1) to little or no training in projects or tools (5). An increase in the education attribute value from 3.0 to 4.0 resulted in a cost increase to $21.6M or a 3.72% increase from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to $20.1M or a 3.36% decrease from the baseline cost. Figure 38 shows the variations in the experience values compared to the support costs.



Decrease	Change	Baseline	Increase	Change
$20,082,888	-3.36%	$20,780,864	$21,554,512	3.72%

Figure 39. Personnel Education Compared to Total Cost.

The second maintenance category addressed was technology. The attributes for maintenance technology affect the performance of defect repair and user support in the post production phases of the software development cycle. The attribute that was identified to have the greatest impact on the support effort/cost was replacement and restructure planning. The range of parameter descriptions/values for replacement and restructure planning is from having automated restructuring service from an outside vendor (1) to no formal replacement or restructure strategy (5). An increase in the replacement and restructure planning attribute value from 3.0 to 4.0 resulted in a cost increase to $21.2M or a 2.09% increase from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to $19.8M or a 4.90% decrease from the baseline cost. Figure 40 shows the variations in the replacement and restructure planning values compared to the total support cost.



Decrease	Change	Baseline	Increase	Change
$19,762,200	-4.90%	$20,780,864	$21,216,080	2.09%

Figure 40. Technology Replacement and Restructuring Planning.

The third maintenance category addressed was process. The attributes for maintenance process category affect the performance of defect repair and user support in the post production phases of the software development cycle. The two attributes that were identified to have the greatest impact on the support effort/cost were field maintenance and customer support. Field maintenance concerns the dispatch of maintenance personnel to user or customer locations to assist in defect repairs and problem identification. The range of parameter descriptions/values for field maintenance is from permanent on-site field maintenance personnel (1) to no field maintenance for the project (5). An increase in the field maintenance attribute value from 3.0 to 4.0 resulted in a cost decrease to $14.8M or a 29.0% decrease from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost increase to $38.2M or an 84.0% increase from the baseline cost. Figure 41 shows the variations in the field maintenance values compared to the total support cost.

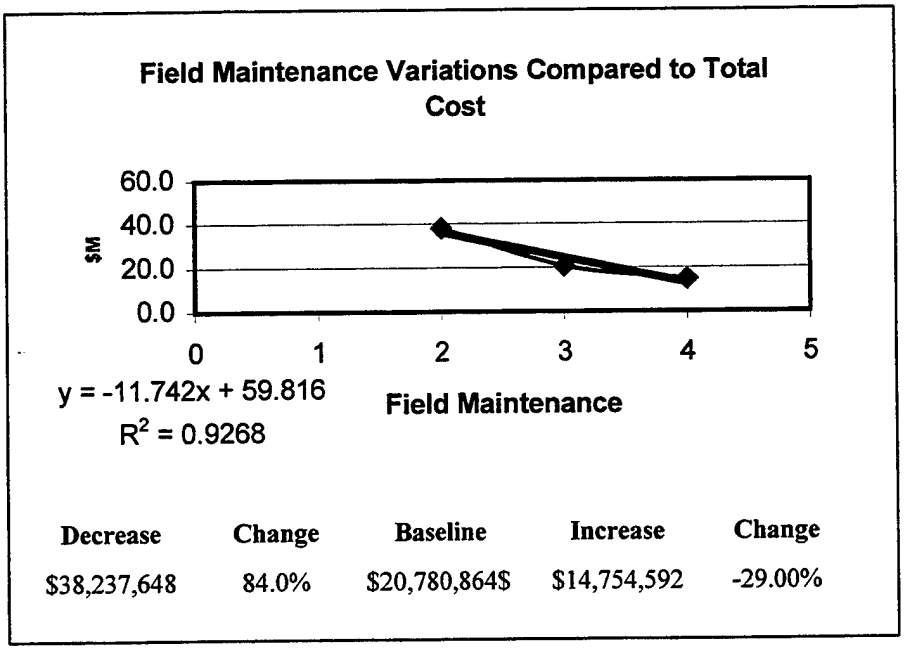
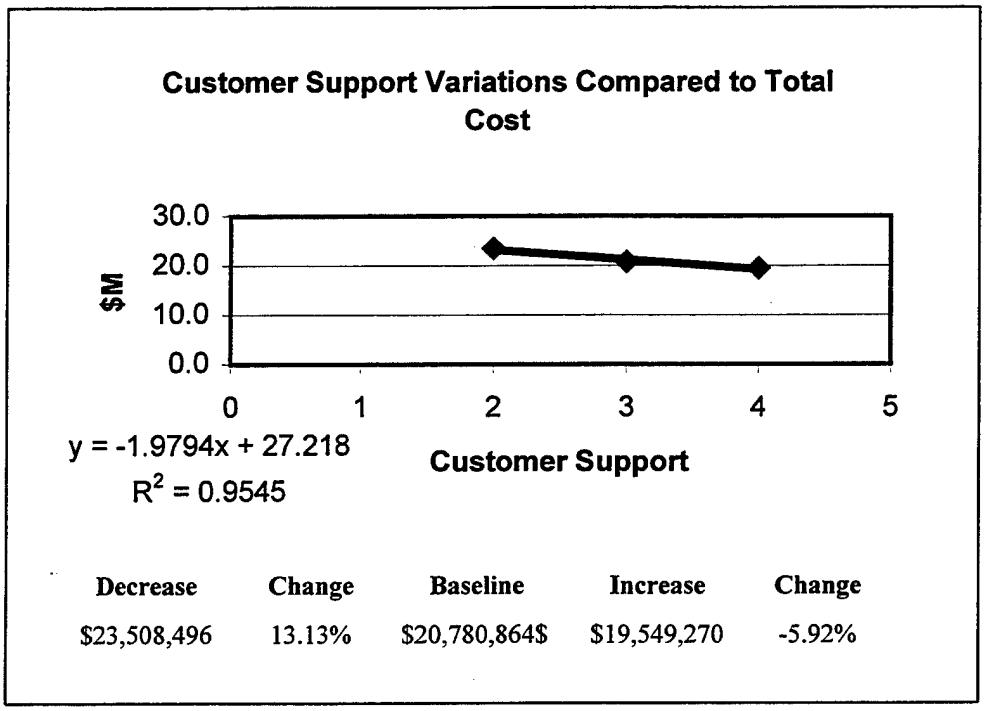


Figure 41. Field Maintenance Compared to Total Cost.

The second maintenance process attribute addressed was customer support. Customer support describes the post installation assistance provided to users. The range of parameter descriptions/values varies from full telephone hot lines with adequate support (1) to limited or no customer support (5). An increase in the customer support attribute value from 3.0 to 4.0 resulted in a cost decrease to $19.5M or a 5.92% decrease from the baseline cost. A decrease in the customer support attribute value from 3.0 to 2.0 resulted in a cost increase to $23.5M or a 13.13% increase from the baseline cost. Figure 42 shows the variations in the customer support values compared to the total support cost.



Decrease	Change	Baseline	Increase	Change
$23,508,496	13.13%	$20,780,864$	$19,549,270	-5.92%

Figure 42. Customer Support Compared to Total Cost.

The fourth maintenance category addressed was environment. Maintenance environment attributes capture the production characteristics for the software product. System quality, performance considerations, and the number of installation sites influence the support effort. Installation and production geography deals with the number of sites where the software will be physically installed on computers. The installation and production geography attribute was identified to have the greatest impact on the support effort/cost in the environment category. The range of parameter descriptions/values for installation and production geography is from a single production site, in a single city (1) to installation and production not being defined (5). An increase in the installation and production geography attribute value from 3.0 to 4.0 resulted in a cost increase to $22.1M or a 6.27% increase from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to $19.7M or a 5.12% decrease from the baseline cost. Figure 43 shows the variations in the installation and production geography values compared to the total support cost.

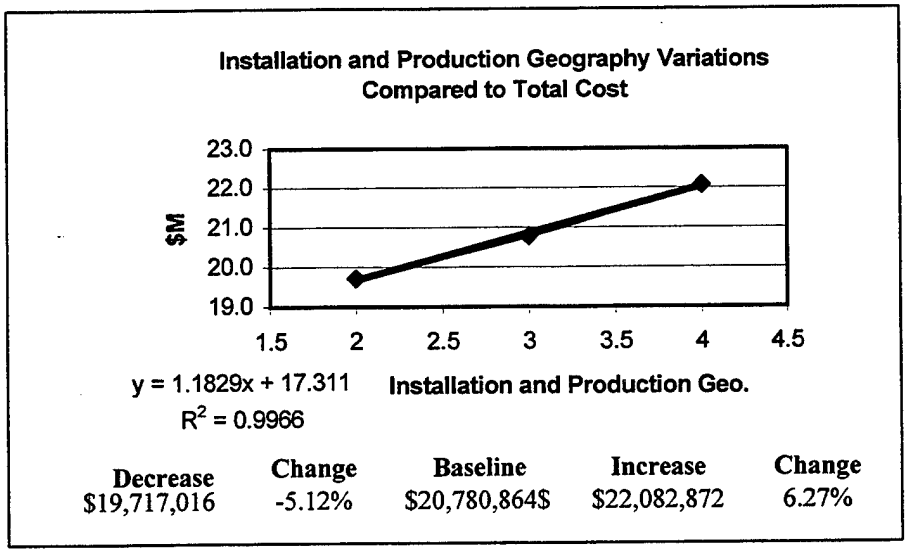


Figure 43. Installation and Production Geography Compared to Total Cost.

The fifth maintenance category addressed was product. The attributes for the maintenance product category capture the production characteristics for the software product. Long range product stability is the volatility of the software over time and the frequency with which new functions, data types, or hardware platforms may be needed, thus affecting the long-range maintenance costs. The range of parameter descriptions/values for long range product stability is from few or no changes to code, data, or to new hardware (1) to frequent changes in functions, data types, and hardware (5). An increase in the long-range product stability attribute value from 3.0 to 4.0 resulted in a cost increase to $21.7M or a 4.06% increase from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to $20.0M or a -3.68% decrease from the baseline cost. Figure 44 shows the variations in the long-range product stability values compared to the total support cost. Table 19 shows the variations in the maintenance attribute values compared to the support costs.

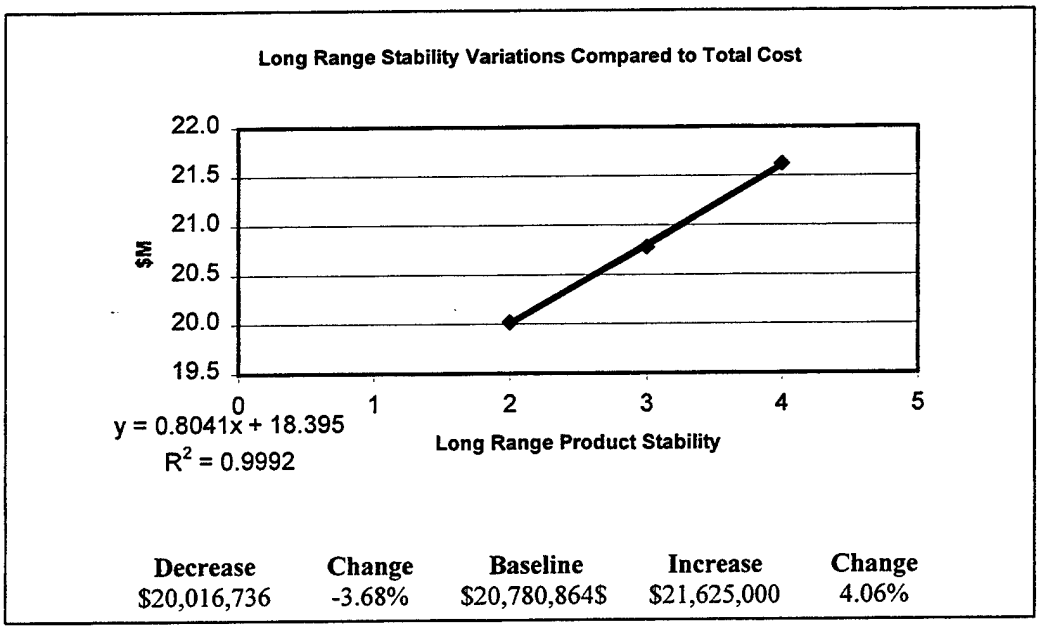


Figure 44. Long Range Product Stability Compared to Total Cost.

Table 19. Maintenance Attribute Variations Compared to Total Cost

KnowledgePLAN 2.0 Build 2026

Software Productivity Research (SPR)

Baseline Support Cost $ 20,780,864

Personnel	**Chg**	**Decrease**	**Change**	**Baseline**	**Chg**	**Increase**	**Change**
Maintenance Personnel STAFFING	3 – 2	20,646,824	-0.65%	20,780,864	3 - 4	20,928,688	0.71%
Maintenance Personnel EXPERIENCE	3 – 2	19,493,824	-6.19%	20,780,864	3 - 4	22,360,608	7.60%
Maintenance Personnel EDUCATION	3 – 2	20,082,888	-3.36%	20,780,864	3 - 4	21,554,512	3.72%
Technology							
Maintenance Platform Computing Support	3 – 2	20,673,840	-0.52%	20,780,864	3 - 4	20,893,168	0.54%
Release Control Methods	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Problem tracking and Reporting	3 – 2	20,675,384	-0.51%	20,780,864	3 - 4	20,890,184	0.53%
Replacement and Restructure Planning	3 – 2	19,762,200	-4.90%	20,780,864	3 - 4	21,216,080	2.09%
Process							
Central Maintenance	3 – 2	20,620,184	-0.77%	20,780,864	3 - 4	20,964,136	0.88%
Field Maintenance	3 – 2	38,237,648	84.00%	20,780,864	3 - 4	14,754,592	-29.00%
Software Warranty Coverage	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Customer Support	3 – 2	23,508,496	13.13%	20,780,864	3 - 4	19,549,720	-5.92%
Delivery Support	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Environment							
Installation and Production Geography	3 – 2	19,717,016	-5.12%	20,780,864	3 - 4	22,082,872	6.27%
Number of System Installation sites	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Annual Growth in Installation Sites (percent)	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Number of System Maintenance Sites	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Product							
Program Execution Frequency	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Current System Status	3 – 2	20,780,864	0.00%	20,780,864	3 - 4	20,780,864	0.00%
Long Range Product Stability	3 – 2	20,016,736	-3.68%	20,780,864	3 - 4	21,625,000	4.06%

Other non-maintenance attributes were found to have an affect on the Knowledge PLAN2.0 model estimate. One non-maintenance attribute was the type of language that the software will be developed in and consequently supported with in the deployment phase. When the type of software was changed from Ada95 to C++, a new total support cost of $19.7M was estimated by the model, which was a decrease of 5.04% from the baseline cost. One note of caution here is that the size of the software will also have to be re-evaluated due to the inherent change/adjustment in the number of lines of code or function points that will be required to develop the software in the new language.

Another attribute category found that affects the support costs is the base code attributes. The first attribute analyzed under this category was base code origin. The base code origin considers where the base code was developed. This attribute takes into consideration that the ability to enhance and maintain a software product. It is considered more difficult if the software was developed by different personnel, with a different methodology, and with different documentation standards than the current organization. An increase in the base origin attribute value from 3.0 to 4.0 resulted in a cost increase to $21.2M or a 2.01% increase from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to $20.4M or a -2.00% decrease from the baseline cost. Table 20 shows the variations in the base code attributes and their corresponding percentage changes.

The second attribute analyzed under this category was base code origin age. The base code origin age considers when the base code was developed. This attribute takes the time period of when the code was developed. Each time period of software development can be characterized by the relative structure design and coding practices,

automated aids used for design, code generation and testing and the availability of skilled personnel. An increase in the base origin age attribute value from 1.0 to 2.0 resulted in a cost increase to $21.1M or a 1.16% increase from the baseline cost. A decrease in the experience attribute value was unable to be performed. Table 20 shows the variations in the base code attributes and their corresponding percentage changes.

The third attribute analyzed under the base code category was base code maintenance responsibility. The base code maintenance responsibility attribute considers who is responsible for the maintenance of the base code that was developed. A formal group external to the organization is more likely to be better focused on the job of maintaining the base code than an internal organization that has other responsibilities. An increase in the base origin maintenance responsibility attribute value from 3.0 to 4.0 resulted in the same cost or no change from the baseline. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to $18.7M or a 9.87% decrease from the baseline cost. Table 20 shows the variations in the base code attributes and their corresponding percentage changes.

The fourth attribute analyzed under this category was base code status. The base code status considers the stabilization of the system. This attribute takes into consideration the overall stability and ease to which modifications to the code can be made. Modification to poorly structured code is unproductive and a difficult task in the software engineering. An increase in the base code status attribute value from 3.0 to 4.0 resulted in a cost increase to $21.7M or a 4.32% increase from the baseline cost. A decrease in the experience attribute value from 3.0 to 2.0 resulted in a cost decrease to

$20.0M or a -3.98% decrease from the baseline cost. Table 20 shows the variations in the base code attributes and their corresponding percentage changes.

Table 20. Base Code Attribute Variations Compared to Total Cost

Base Code Attributes	Decrease/New Total Cost	Change in Cost	Baseline Total Cost	Increase/New Total Cost	Change in Cost
Base Code Origin	20,365,400	-2.00%	20,780,864	21,217,432	2.10%
Base Code Age	Unable to Dec.	0.00%	20,780,864	21,114,016	1.60%
Maintenance Responsibility	18,729,232	-9.87%	20,780,864	20,780,864	0.00%
Stabilizing System	19,954,312	-3.98%	20,780,864	21,677,984	4.32%

In summary, the maintenance support category attributes varied the support effort/costs estimates from no change in the estimate to an 84% change. Thus, care should be exercised when the maintenance category attribute values are decided upon, due to the significant changes in the estimate values. There were also non-maintenance attributes that affected the estimated effort/cost. These attributes ranged from the type of language that the software was to be developed in to the base code characteristics. The complexity category located in the sizing summary is subdivided into base code, new code, and update code tabs with further refinement into problem, code, and data for each code type. After making numerous adjustments in their values they were found not to have an effect on the support estimate.

KnowledgePLAN's definition of maintenance is that maintenance is comprised of the activities performed to support or manage a software environment. The following task categories can be found under the maintenance estimate: customer support, field service, central maintenance, maintenance management, and system deployment.

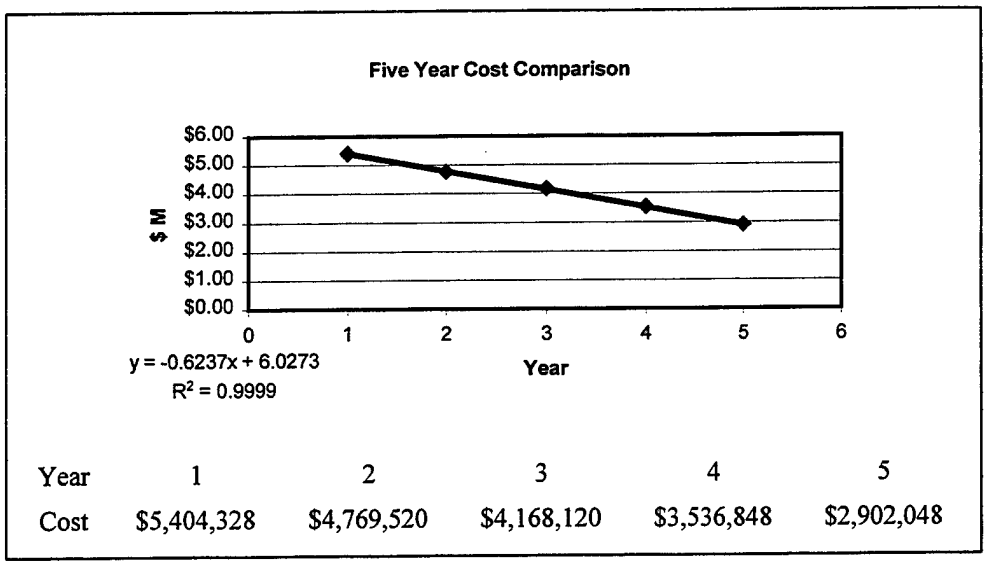
Customer support refers to a number of different tasks that include answering telephone inquiries, providing training for new users, and relaying user defect reports to the central maintenance facility.

Field service refers to on-site assistance given to customers at their own locations. Field service is normally associated with large commercial software packages. Maintenance management refers to the normal supervisory tasks associated with managing technical staff, performing supervisor duties, handling budgets, expense tracking, and other management related activities.

Central maintenance is a maintenance category where the defect rework is accounted for in the estimate. Central maintenance is sub-divided into maintenance defect rework preparation, maintenance defect rework execution, and maintenance defect rework repair. Maintenance defect rework preparation is the task category related to administration of the software maintenance environment. This includes tool acquisition and education for the maintenance engineers on the system. Maintenance defect rework execution includes the analysis and prioritization of defect reports in a software maintenance environment, where the individuals reporting defects may be customers or internal personnel working on software enhancements. Maintenance defect rework repair consists of the effort needed to fix defects found in a legacy system or released software product. Other work that is included in this category of central maintenance is the effort needed to debug and test the fixes made to the software.

System deployment is the configuration of a software application for use in a specific user's environment. The activities associated with system deployment include setting flags, options, and other non-programming adjustments to an application.

The support costs for the 5-year period were graphed against each year to give a representation of how the cost were estimated over the time period. Figure 45 shows the results of the graph with the corresponding values for each year.



Year	1	2	3	4	5
Cost	$5,404,328	$4,769,520	$4,168,120	$3,536,848	$2,902,048

Figure 45. Five Year Cost Comparison.

Table 21 provides the cost category breakout for the five year support period and table 22 shows the SPR KnowledgePLAN database composition.

Table 21. Estimate Overview by Category.

Cost Category Breakout for the 5 Year Support Period

Task Category	Start	Finish	Plan FTE	Plan Work	Plan Cost	Plan Deliverable	Defect Removal Efficiency
System Deploy	6/25/98	7/8/98	3.00	01.58 M	$23,960*	3555.00fp	0%
MaintDef RepPrep	7/8/98	7/10/03	0.49	33.71 M	$512,344	175.00 kl	0%
MaintDef RepExec	7/8/98	7/10/03	0.94	64.60 M	$981,992	175.00 kl	11%
MaintDef RepRepair	7/8/98	7/10/03	2.76	190.06 M	$2,888,856	1023.00 df	0%
Field Service	7/8/98	7/10/03	11.13	765.30 M	$11,632,544	3557.00fp	0%
Customer Support	7/8/98	7/10/03	0.96	66.11 M	$1,004,800	3557.00fp	0%
Maint Mgmt	7/8/98	7/10/03	3.60	247.39 M	$3,760,328	123.00 pe	0%
5-Year Total					**$20,780,864**		

* Note that deployment costs are not included in the 5 year total estimate

Table 22. SPR KnowledgePLAN Database Composition.

Size of Project in function Points	MIS and Outsource	Sys. and Embed.	Military and Defense	Commercial	Other	**Total**	Percent
Very Small <10	90	130	25	15	35	**295**	4%
Small 11-100	450	275	35	80	150	**990**	12%
Low Medium 101-250	775	600	65	130	300	**1,870**	22%
Medium 251-1000	1050	925	30	125	400	**2,530**	30%
Large 1001-5000	450	700	75	175	300	**1,700**	20%
Very Large 5001-20000	250	350	60	50	150	**860**	10%
Super Large >20000	70	85	15	15	10	**195**	2%
Total	**3,135**	**3,065**	**305**	**590**	**1,345**	**8,440**	100%
Percent	*37%*	*36%*	*4%*	*7%*	*16%*	*100%*	

The main types of projects in the database were MIS, Outsource, Systems, and Embedded type projects. The size of the systems was from ten lines of code to greater than 20,000 lines of code. The majority fell in the low-medium, medium, and large size ranges.

The 20-years of support costs that were looked at for the five models showed varying methods of how the costs were estimated and allocated the 20-years throughout the support effort. The PRICE-S had made changes in the support cost for every year of the twenty-year estimate (21-years to account for the partial first and twentieth year in order to get a whole year's worth of support effort). The SEER-SEM and SoftEst models made changes in the yearly values for the first four years and then flat lined thereafter to

year twenty. SEER-SEM also had 21-years in the estimate in order to account for the partial years to make a full twenty years. The SoftCost-OO made changes in the yearly values for the first three years and then flat lined to year twenty. KnowledgePLAN 2.0 had made changes in the support costs for every year of the five-year estimate. Table 23 provides a 20 year support cost comparison for the given models, with the exception of SPR KnowledgePLAN, which only estimates a five year support period. Figure 46 provides a graphical representation of the 20 year support cost comparison of the models.

Table 23. 20-Year Support Costs - a Comparison of the Models.

Year	SEER-SEM	Softcost-OO	PRICE-S	SoftEst	Knowledge-PLAN
1	$9,944,596	$3,402,100	$2,041,360	$6,529,209	$5,404,328
2	$12,034,101	$3,118,600	$3,009,600	$5,658,648	$4,769,520
3	$6,371,501	$2,835,000	$3,078,000	$5,005,726	$4,168,120
4	$4,289,502	$2,835,000	$3,082,560	$4,352,806	$3,536,848
5	$4,289,502	$2,835,000	$3,053,680	$4,352,806	$2,902,048
6	$4,289,502	$2,835,000	$2,971,600	$4,352,806	
7	$4,289,502	$2,835,000	$2,830,240	$4,352,806	
8	$4,289,502	$2,835,000	$2,632,640	$4,352,806	
9	$4,289,502	$2,835,000	$2,390,960	$4,352,806	
10	$4,289,502	$2,835,000	$2,120,400	$4,352,806	
11	$4,289,502	$2,835,000	$1,831,600	$4,352,806	
12	$4,289,502	$2,835,000	$1,536,720	$4,352,806	
13	$4,289,502	$2,835,000	$1,254,000	$4,352,806	
14	$4,289,502	$2,835,000	$984,960	$4,352,806	
15	$4,289,502	$2,835,000	$743,280	$4,352,806	
16	$4,289,502	$2,835,000	$535,040	$4,352,806	
17	$4,289,502	$2,835,000	$363,280	$4,352,806	
18	$4,289,502	$2,835,000	$237,120	$4,352,806	
19	$4,289,502	$2,835,000	$148,960	$4,352,806	
20	$4,289,502	$2,835,000	$95,760	$4,352,806	
21	$1,537,067		$6,080		
Total	$101,271,732	$57,550,700	$34,941,760	$91,191,285	$20,780,864

Note: Years 1 and 21 are only partial years for SEER-SEM and PRICE-S due to support effort start dates.

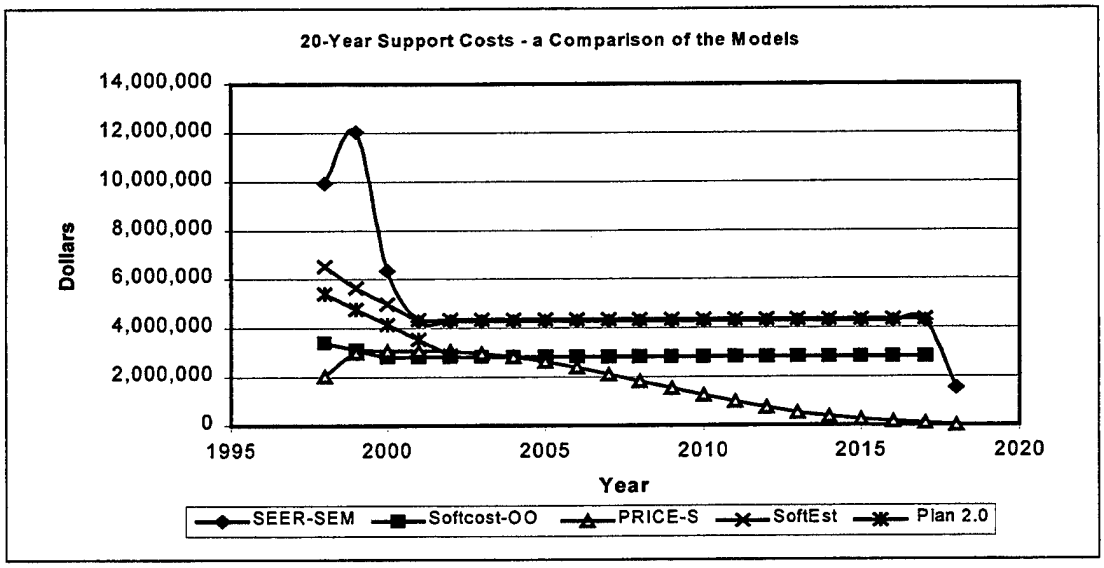


Figure 46. 20-Year Support Costs - a Comparison of the Models.

The software support estimating models were then compared to one another in terms of the total estimate for the 20-year support period. The model with the highest estimate was SEER-SEM at $101.3M, which was 290% greater than the estimate generated by PRICE-S and only 111% greater than the SoftEst. The second highest estimate was SoftEst at $91.2M, which was 261% greater than PRICE-S and 158% greater than SoftCost-OO. The third highest estimate was SoftCost-OO at $57.6M, which was 165% higher than PRICE-S and 158% higher than SoftEst. PRICE-S at $34.9M, was the lowest estimate. SPR KnowledgePLAN only included a template for a 5 year support period. Table 24 provides a summary of the comparison of the models. This comparison of the models was only tested for the given base case in this research effort and should not be applied in the field of software cost estimating without further study and analysis.

Table 24. Model Estimates Compared to Other Models in Percentage Terms

Model Comparison for 20 Years of Support						
Estimate	$M	34.9	101.3	57.6	91.2	N/A
		PRICE-S	SEER-SEM	SoftCost-OO	SoftEst	SPR Knowledge PLAN
PRICE-S	34.9	100.00%	290.26%	165.04%	261.32%	N/A
SEER-SEM	101.3	34.45%	100.00%	56.86%	90.03%	N/A
SoftCost-OO	57.6	60.59%	175.87%	100.00%	158.33%	N/A
SoftEst	91.2	38.27%	111.07%	63.16%	100.00%	N/A
SPR Knowledge Plan	N/A	N/A	N/A	N/A	N/A	N/A

V. Conclusions and Recommendations

Overview

The purpose of this research effort was to develop a consolidated document which highlights the differences in definitions, assumptions, methodologies, and theory used by PRICE-S, SEER-SEM, SoftCost-OO, SoftEst, and SPR KnowledgePLAN cost models and to examine what the impact of these differences have on the resulting support estimates. Conclusions regarding this effort are constrained by the research objectives outlined in Chapter I and in Appendix B. Although numerous differences between the models were identified, this research does not cover all software support issues and therefore should not be used as a comprehensive source. For the other issues that were not addressed in this effort, the researchers make several recommendations for further research in the software support arena.

Conclusion

Five main research objectives guided this research effort. The conclusions will be addressed in context of these objectives.

Research Objective 1. What general support activities and specific cost elements are estimated? The common theme among the models was maintenance, enhancement, and growth in the estimation of the support costs. All of the models touched upon one or more of the categories. However, not all models reported all the costs at the sub-category level. The models also used different names for the various sub-categories of support and had have different input parameters required for the support and development efforts. Each model used different algorithms or a combination of algorithms for estimating software support effort/cost. The specific cost elements for software support are model

dependent. The cost for documentation was considered for all the models either directly or indirectly within another cost category. The support cost for program management was once again directly or indirectly identified by all the models with the exception to SoftEst, which only considered error corrections/bug fixes and small enhancements to the software.

Research Objective 2. How does each model define a source line of code and how are language differences accounted for? The models used a fairly consistent definition for the source lines of code used or the given number of function points to be used in place of lines of code. All of the models allowed SLOC to be entered into the model for generating the estimate, whether SLOC was directly used or converted into Function Points through a conversion equation. Note that input parameters within the development or support environment were found to have made an adjustment to the number of lines of code that the model actually uses to perform the given estimate. Thus, changes to the initial SLOC input by the user occurred within the model. The models accounted for the differences in the programming languages by adjusting the model's estimate according to the change in language made. For example, a change in the language parameter for the PRICE-S model in CSCI 2 (C++ to Ada95) resulted in a 17% increase in the support effort estimate. A change in the language parameter in the SoftEst model resulted in a change in the coefficients and exponents in the estimation equations, which resulted in a change of the estimate.

Research Objective 3. Which model factors are the key cost drivers for support cost? What estimating methodology was used to develop support factors? Unfortunately, these questions cannot be answered in a general nature. The models had

numerous support factors that varied the support cost estimate. There was no single cost driver or methodology that was used by any of the models in the estimation of the effort or cost. In some models, a single change in a development input parameter would result in a dramatic change in the support estimate. The range of percentage changes between the models was seen from a high of 84% to a low of a .04%. Other models had very limited changes in support when adjusting the development inputs. Some of the unique support factors identified by the model designers had no effect on the resulting support estimate. Once again, the models varied in regards to which parameters caused a change that a consolidated conclusion could not be reached.

Research Objective 4. How do the models account for variations in the number of years a system is deployed? All the models researched allowed for different periods of support. SPR KnowledgePLAN, however, was limited to a five-year template and therefore, a five-year support estimate. All models showed basically the same trend, higher support cost in the initial years and then a leveling off or a continual small decrease later in the life cycle, with the exception of PRICE-S, which continually declines. The models all used different algorithms depending on the number of years input.

Research Objective 5. Are there any distinctive characteristics about the data base(s) used by the various models? The databases used by the various models were also extremely varied. Some had large amounts of military avionics with little MIS, while others were just the opposite. The number of languages used in the databases was also largely varied. This information was dependent on the amount of information that the developer had on a given company depending on the model developer.

The researchers feel a typical user would not be able to normalize the various software support models or generalize the types of inputs required. This is due to the inherent definitions and the amount of importance placed upon the input parameters in the cost estimation methodology of the various models. Users should concentrate on learning two of the models very well, which will allow for running cross checks. The appropriate model to use is dependent on the project or system being estimated since the databases are varied between the software support models.

Recommendations

This research effort did not touch upon all aspects of the software support cost estimating process. The researchers did find numerous issues that could be addressed in future research efforts. The first area is that additional research should be performed in attempting to calibrate the support cost estimating models to military software databases and then perform a verification and validation of the models to the various projects.

The second area of additional research would be to determine the accuracy of information that is located in the databases for the various models. This would include determining the level of cost information (program, project, CSCI, or CSC level) in the database. This area of concern maybe addressed by both the commercial and government sectors.

The third area would be to determine a (best practices) set of attributes or parameters that could be used to represent the support effort that would in turn capture the costs associated with performing the support to aid in the development of an accurate estimate of the support costs. An activity based cost estimation of the support effort may

be beneficial or a survey of the maintenance functions being performed and the percentage of effort spent on each category.

The fourth area of additional research would be to develop a set of generic parameters or attributes that could be translated into the definitions of the attributes for the various models, so that the inputs could be generalized for the models.

In conclusion, this research effort identified many key similarities and differences between five software support cost estimating models. The researchers' feel that the differences in definitions for model input parameters, internal algorithms, and key assumptions about the different types of software support activities make it nearly impossible for a model user to normalize the different models. The researchers desire that this effort resulted in a useful, consolidated document, which gives practical knowledge about the field of software support, and helps model users understand why the models produce different estimates for an identical type of software design and support environment.

Appendix A: Baseline Case Used for the Models

	REVIC	SEER-SEM	PRICE-S	SoftCost-OO	SPR KnowledgePLAN
CSCI 1	Separate Data file for each CSCI 2 CSCs loaded into this file CSC1: 20K SLOC CSC2: 30K SLOC ADA Embedded Mode RELY: HI Default values for all other inputs	One Data File for Total Estimates 2CSCs CSC1: 20K SLOC CSC2: 30K SLOC Avionics Platform Flight Application Ada Development Method 2167A min Development Standard Default values for all other inputs	One Data File for Total Estimate 2 CSCs CSC1: 20K SLOC CSC2: 30K SLOC PROFAC: 5.00 APPL: 5.50 PLTFM: 1.80 SSR Date: 894 INTEGE & INTEGI: 0.50 CPLX1: 1.00 Default values for all other inputs	One Data File for Total Estimate 2 CSCs CSC1: 20K SLOC CSC2: 30K SLOC Avionics (Military) Centralized 1 Organization High Degree of Standardization 1 OO Project Completed Nominal values for all other inputs	One Data File for Total Estimate 1 CSCI : 175 K SLOC Prog: Appl Standalone Gov. Mil. Cont. Sys: Embedded Metric Sizing: Lang: Ada 95 Base Code Age: New or less than year Base Code Origin: Developed under custom contract Default values for all other inputs
CSCI 2	Separate Data File Embedded Dev Mode PCAP:HI LEXP: HI RELY: HI DT&E: 28% 80K SLOC Default values for all other inputs	Avionics Platform Flight Application Waterfall Development Method 2167A min Development Standards Programmer Capability: Nom, Hi, VHI Programmer Lang Exp: HI, VHI, EHI Language Type: HI, HI, VHI 80K SLOC Default values for all other inputs	PROFAC: 5.00 APPL: 5.50 PLTFM: 1.80 SSR Date: 894 INTEGE & INTEGI: 0.50 CPLX1: 0.80 Non-Executable SLOC: 0 Default values for all other inputs	80K SLOC Avionics (Military) Centralized 1 Organization High Degree of Standardization 1 OO Project Completed Nominal values for all other inputs	
CSCI 3	Separate Data File ADA Development Mode RELY: HI 45K CLOC Default values for all other inputs	Avionics Platform Flight Application ADA Development Method 2167A Min Development Standard 45K SLOC Default values for all other inputs	PROFAC: 5.00 APPL: 5.50 PLTFM: 1.80 SSR Date: 894 INTEGE & INTEGI: 0.50 CPLX1: 1.00	45K SLOC Avionics (Military) Centralized 1 Organization High Degree of Standardization 1 OO Project Completed Nominal values for all other inputs	

Appendix B: Checklist Used to Examine Cost Models

1. What general support activities and specific cost elements are estimated?

a. What general support activities are included in the model estimates?

b. What specific cost elements are estimated by the model? What do they mean or represent?

c. Does the model include the cost of updating the documentation, is that cost separately identified?

d. Does the model include the cost of program management, if so, is the cost separately identified?

2. How does each model define a source line of code and how are language differences accounted for?

a. What is each model's definition of source lines of code?

b. Do the models account for language differences?

3. Which model factors are key cost drivers for support costs? What estimating methodology was used to develop support factors?

a. Do model development factors affect support costs. To what extent do these factors affect the support costs.

b. Identify support unique cost drivers used by each model to develop estimates (i.e which factors have the most significant impact on development effort?).

c. How were these support factors developed? What estimating methodology was used? Linear regression or some other statistical method? Expert Judgment? Heuristics? Composite?

4. How does the model account for variations in the numbers of years a system is deployed?

a. Does the model allow for variations in the number of years to be supported?

b. What, if any, penalties are assessed when the schedule is compressed or extended?

c. What estimating methodology was used? Linear regression or some other statistical method? Expert Judgment? Heuristics? Composite?

5. Are there any distinctive characteristics about the data base(s) used by the various models?

a. How many projects were in the database used to develop the model?

b. Did it have any unique characteristics? If yes, how were these special characteristics normalized in developing the generic model?

c. How much of the database was military systems versus commercial systems? Embedded systems versus MIS systems?

d. What programming languages were included in the database (% each)?

e. What was the distribution of the records in the database by size (project level, CSCI, CSC, CSU)?

Appendix C: PRICE-S Reports

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Tuesday June 23 1998 TIME 12:44 PM Project : PRICENEW
394028

CSCI 1 Devt. Item w/comps

ITEM DESCRIPTORS
Platform 1.80 Mgmt Complexity 1.00 External Integ 0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Spec. Review	0
Pre. Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

COMPONENT 1 titled: CSC 1
DESCRIPTORS
Internal Integ 0.50 External Integ 0.50
Utilization Fraction 0.50

SCHEDULE

Software Spec. Review	894	Pre. Design Review	0
Critical Design Review	0	Test Readiness Review	0
Functional Config Audit	0		

LANGUAGE 1 DESCRIPTORS
Language Ada95 Source Code 20000 Non-executable SLOC 0.00
Complexity 1 1.00 Complexity 2 1.00 Productivity Factor 5.000
Application 5.50 New Design 1.00 New Code 1.00

Application Categories	Mix	New Design	New Code
User Defined (APPL = 5.50)	1.00	1.00	1.00
DATA S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Operating Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Tuesday June 23 1998 TIME 12:44 PM Project : PRICENEW
394028

COMPONENT 2 titled: CSC 2
DESCRIPTORS
Internal Integ 0.50 External Integ 0.50
Utilization Fraction 0.50

SCHEDULE
Software Spec. Review 894 Pre. Design Review 0
Critical Design Review 0 Test Readiness Review 0
Functional Config Audit 0

LANGUAGE 1 DESCRIPTORS
Language Ada95 Source Code 30000 Non-executable SLOC 0.00
Complexity 1 1.00 Complexity 2 1.00 Productivity Factor 5.000
Application 5.50 New Design 1.00 New Code 1.00

Application Categories	Mix	New Design	New Code
User Defined (APPL = 5.50)	1.00	1.00	1.00
DATA S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Operating Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Tuesday June 23 1998 TIME 12:44 PM Project : PRICENEW
394028

CSCI 2 Development Item

ITEM DESCRIPTORS

Platform	1.80	Mgmt Complexity	1.00	Cost	0.00
Internal Integ	0.500	External Integ	0.500	Utilization	0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Spec. Review	894
Pre. Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language C++	Source Code	80000	Non-executable SLOC	0.00	
Complexity 1	0.80	Complexity 2	1.00	Productivity Factor	5.000
Application	5.50	New Design	1.00	New Code	1.00

Application Categories	Mix	New Design	New Code
User Defined (APPL = 5.50)	1.00	1.00	1.00
DATA S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Operating Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Tuesday June 23 1998 TIME 12:44 PM Project : PRICENEW
394028

CSCI 3 Development Item

ITEM DESCRIPTORS

Platform	1.80	Mgmt Complexity	1.00	Cost	0.00
Internal Integ	0.500	External Integ	0.500	Utilization	0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Spec. Review	894
Pre. Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language	Ada95	Source Code	45000	Non-executable SLOC	0.00
Complexity 1	1.00	Complexity 2	1.00	Productivity Factor	5.000
Application	5.50	New Design	1.00	New Code	1.00

Application Categories	Mix	New Design	New Code
User Defined (APPL = 5.50)	1.00	1.00	1.00
DATA S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Operating Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode

DATE Tuesday June 23 1998 TIME 12:45 PM Project : PRICENEW
394028

Price S Model

SYSTEM SUMMARY TOTALS

Costs in Person Months

	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
Design Engineering	176.8	269.8	187.9	634.5
Programming	117.8	222.3	119.1	459.2
Data	95.4	114.5	67.2	277.1
Systems Eng/Prog Mgmt	103.0	180.8	128.3	412.1
Quality Assurance	96.2	98.5	60.0	254.7
Configuration Control	96.2	98.5	66.8	261.5
TOTAL	685.4	984.5	629.2	2299.1

Acquisition Costs	6899.8
Modification Costs	0.0
Life Cycle Costs	2299.1
Total Cost	9199.0

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode

DATE Tuesday June 23 1998 TIME 12:46 PM Project : PRICENEW
394028

CSCI 1 Devt. Item w/comps

Global Title: Uses System Globals
Escalation Title: Uses System Escalation
Financial Title: Uses System Financials
Deployment Title: Uses System Deployment

Costs in Person Months

20.1 YEAR OPERATIONAL LIFE

	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
Design Engineering	52.7	83.8	56.1	192.5
Programming	35.1	70.3	36.3	141.8
Data	27.7	35.4	19.9	83.0
Systems Eng/Prog Mgmt	30.3	55.5	37.8	123.6
Quality Assurance	28.1	31.0	18.1	77.2
Configuration Control	28.1	31.0	20.1	79.2
TOTAL	202.0	307.0	188.4	697.4

Summary
55000 Source Lines as of (JAN 18)

Initial Defects/KSLOC :	1.5
Acquisition Costs	2109.1 *
Life Cycle Costs	697.4
TOTAL	2806.5

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode

DATE Tuesday June 23 1998 TIME 12:46 PM Project : PRICENEW
394028

CSCI 2 Development Item

Global Title: Uses System Globals
Escalation Title: Uses System Escalation
Financial Title: Uses System Financials
Deployment Title: Uses System Deployment

Costs in Person Months

20.1 YEAR OPERATIONAL LIFE

	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
Design Engineering	77.1	110.1	81.8	269.1
Programming	51.4	88.1	50.2	189.7
Data	43.3	47.5	29.6	120.4
Systems Eng/Prog Mgmt	45.8	75.4	56.8	178.0
Quality Assurance	43.3	39.7	25.8	108.7
Configuration Control	43.3	39.7	28.8	111.7
TOTAL	304.2	400.4	273.0	977.7

Summary
88000 Source Lines as of (JAN 18)

Initial Defects/KSLOC :	1.8
Acquisition Costs	2542.2 *
Life Cycle Costs	977.7
TOTAL	3519.9

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode

DATE Tuesday June 23 1998 TIME 12:46 PM Project : PRICENEW
394028

CSCI 3 Development Item

Global Title: Uses System Globals
Escalation Title: Uses System Escalation
Financial Title: Uses System Financials
Deployment Title: Uses System Deployment

Costs in Person Months

20.1 YEAR OPERATIONAL LIFE

	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
Design Engineering	47.0	75.9	50.0	172.8
Programming	31.3	63.9	32.5	127.7
Data	24.4	31.7	17.7	73.7
Systems Eng/Prog Mgmt	26.9	50.0	33.7	110.5
Quality Assurance	24.8	27.8	16.1	68.7
Configuration Control	24.8	27.8	17.8	70.5
TOTAL	179.2	277.1	167.8	624.1

Summary
49500 Source Lines as of (JAN 18)

Initial Defects/KSLOC :	1.5
Acquisition Costs	1786.1 *
Life Cycle Costs	624.1
TOTAL	2410.2

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode
CSCI 1 Development Item
ANNUAL ACTIVITY Costs in Person Months

YEAR	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
1998	33.4	6.9	2.6	42.90
1999	36.2	17.9	7.1	61.20
2000	25.5	25.4	10.7	61.60
2001	18.1	29.9	13.3	61.30
2002	13.7	31.9	15.2	60.80
2003	10.9	32.0	16.2	59.10
2004	9.1	30.6	16.6	56.30
2005	7.9	28.0	16.5	52.40
2006	7.0	24.8	15.8	47.60
2007	6.2	21.2	14.8	42.20
2008	5.6	17.4	13.4	36.40
2009	5.0	13.7	11.7	30.40
2010	4.5	10.3	10.0	24.80
2011	4.0	7.3	8.1	19.40
2012	3.5	4.8	6.3	14.60
2013	3.0	2.8	4.6	10.40
2014	2.6	1.4	2.9	6.90
2015	2.2	0.6	1.6	4.40
2016	1.9	0.2	0.7	2.80
2017	1.6	0.0	0.1	1.70
2018	0.1	0.0	0.0	0.10
TOTAL:	202.00	307.10	188.20	697.30

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode
CSCI 2 Development Item
ANNUAL ACTIVITY Costs in Person Months

YEAR	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
1998	38.7	9.0	3.8	51.5
1999	47.7	23.4	10.4	81.5
2000	37.4	33.1	15.5	86.0
2001	28.5	39.0	19.3	86.8
2002	22.4	41.6	22.0	86.0
2003	18.4	41.7	23.5	83.6
2004	15.6	39.9	24.1	79.6
2005	13.5	36.6	23.9	74.0
2006	11.9	32.4	22.9	67.2
2007	10.6	27.6	21.4	59.6
2008	9.5	22.7	19.4	51.6
2009	8.6	17.8	17.1	43.5
2010	7.7	13.4	14.5	35.6
2011	6.8	9.5	11.8	28.1
2012	6.1	6.2	9.1	21.4
2013	5.3	3.7	6.6	15.6
2014	4.7	1.9	4.3	10.9
2015	4.1	0.8	2.4	7.3
2016	3.5	0.2	1.0	4.7
2017	3.0	0.0	0.2	3.2
2018	0.2	0.0	0.0	0.2
TOTAL	304.2	400.5	273.2	977.9

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode
CSCI 3 Development Item
ANNUAL ACTIVITY Costs in Person Months

YEAR	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
1998	31.40	6.20	2.30	39.90
1999	32.80	16.20	6.30	55.30
2000	22.50	22.90	9.50	54.90
2001	15.80	27.00	11.90	54.70
2002	11.80	28.80	13.50	54.10
2003	9.40	28.90	14.50	52.80
2004	7.90	27.60	14.80	50.30
2005	6.80	25.30	14.70	46.80
2006	6.00	22.40	14.10	42.50
2007	5.40	19.10	13.20	37.70
2008	4.90	15.70	11.90	32.50
2009	4.40	12.30	10.50	27.20
2010	3.90	9.30	8.90	22.10
2011	3.40	6.60	7.30	17.30
2012	3.00	4.30	5.60	12.90
2013	2.60	2.60	4.00	9.20
2014	2.20	1.30	2.60	6.10
2015	1.90	0.50	1.50	3.90
2016	1.60	0.10	0.60	2.30
2017	1.30	0.00	0.10	1.40
2018	0.10	0.00	0.00	0.10
TOTAL	179.1	277.1	167.8	624.0

--- PRICE SOFTWARE MODEL ---
Life Cycle Mode
PRICE-S Total System Support Costs for Twenty Years
ANNUAL ACTIVITY Costs in Person Months

YEAR	MAINTENANCE	ENHANCEMENT	GROWTH	TOTAL
1998	103.50	22.10	8.70	134.30
1999	116.70	57.50	23.80	198.00
2000	85.40	81.40	35.70	202.50
2001	62.40	95.90	44.50	202.80
2002	47.90	102.30	50.70	200.90
2003	38.70	102.60	54.20	195.50
2004	32.60	98.10	55.50	186.20
2005	28.20	89.90	55.10	173.20
2006	24.90	79.60	52.80	157.30
2007	22.20	67.90	49.40	139.50
2008	20.00	55.80	44.70	120.50
2009	18.00	43.80	39.30	101.10
2010	16.10	33.00	33.40	82.50
2011	14.20	23.40	27.20	64.80
2012	12.60	15.30	21.00	48.90
2013	10.90	9.10	15.20	35.20
2014	9.50	4.60	9.80	23.90
2015	8.20	1.90	5.50	15.60
2016	7.00	0.50	2.30	9.80
2017	5.90	0.00	0.40	6.30
2018	0.40	0.00	0.00	0.40
TOTAL:	685.30	984.70	629.20	2299.20

Appendix D: SEER-SEM Report

SEER-SEM Baseline Inputs

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Vers. 4.52d

Project: SEER-SEM CSCI: CSCI 2 **Inputs** 6/23/98 12:17:31 PM

	Least	Likely	Most
LINES			
New Lines of Code	**80,000**	**80,000**	**80,000**
Pre-exists, not designed for reuse	0	0	0
Pre-existing lines of code	0	0	0
Lines to be deleted in pre-exstg	0	0	0
Redesign required	5.00%	10.00%	40.00%
Reimplementation required	1.00%	5.00%	10.00%
Retest required	10.00%	40.00%	100.00%
Pre-exists, designed for reuse	0	0	0
Function Implementation Mechanism		**C++**	
CSCIs Included In Size	1	1	1
PERSONNEL CAPABILITIES & EXPERIENCE			
Analyst Capabilities	Low	Nom	Hi
Analyst's Application Experience	Nom-	Nom	Nom+
Programmer Capabilities	**Nom**	**Hi**	**VHi**
Programmer's Language Experience	**Hi**	**VHi**	**EHi**
Host System Experience	Nom	Hi	VHi
Target System Experience	Nom	Hi	Hi
Practices & Methods Experience	Nom	Hi	VHi
DEVELOPMENT SUPPORT ENVIRONMENT			
Modern Development Practices Use	Nom-	Nom	Nom+
Automated Tools Use	Nom	Nom+	Hi
Logon thru Hardcopy Turnaround	VLo	Low-	Nom
Terminal Response Time	Low-	Hi-	Hi
Multiple Site Development	Nom	Nom	Nom
Resource Dedication	Nom	Nom	Nom
Resource and Support Location	Nom	Nom	Nom
Host System Volatility	Nom	Nom	Nom
Process Volatility	Low	Low+	Nom
PRODUCT DEVELOPMENT REQUIREMENTS			
Requirements Volatility (Change)	Nom	Nom	Hi
Specification Level – Reliability	Hi+	Hi+	VH-
Test Level	Hi+	Hi+	VH-
Quality Assurance Level	Hi-	Hi+	VHi
Rehost from Development to Target	Nom	Hi	VHi+
PRODUCT REUSABILITY REQUIREMENTS			
Reusability Level Required	Nom	Nom	Nom
Software Impacted by Reuse	100.00%	100.00%	100.00%
DEVELOPMENT ENVIRONMENT COMPLEXITY			
Language Type (complexity)	**Hi**	**Hi**	VHi
Host Development System Complexity	Nom	Nom	Nom
Application Class Complexity	Hi-	Hi	Hi
Process Improvement	Nom	Nom	Nom
TARGET ENVIRONMENT			
Special Display Requirements	Nom	Hi	VHi
Memory Constraints	Hi-	Hi	Hi+
Time Constraints	Nom	Nom+	Hi-
Real Time Code	Nom	Nom+	Hi
Target System Complexity	Nom	Nom	Nom
Target System Volatility	Hi-	Hi	Hi+

SEER-SEM Baseline Inputs Continued

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Vers. 4.52d

Project: SEER-SEM CSCI: CSCI 2	Inputs	6/23/98	12:17:31	PM
		Least	Likely	Most
Security Requirements		Nom	Nom	Nom
SCHEDULE & STAFFING CONSIDERATIONS				
Required Schedule (Calendar Mos)				
Start Date			**8/01/94**	
Complexity (Staffing)		VHi-	VHi	VHi+
Staff Loading			Nom	
Min Time vs. Opt Effort			Minimum Time	
STAFFING CONSTRAINTS		Start Month	Min Staff	Max Staff
Staff Level (next)		0	0.00	0.00
RISK ANALYSIS				
Effort Probability			50.00%	
Schedule Probability			50.00%	
REQUIREMENTS				
Requirements Complete at Start			Low	
Requirements Definition Formality		Nom	Nom+	Hi+
Requirements Effort After Baseline			YES	
SYSTEM INTEGRATION				
CSCIs Concurrently Integrating			1	
Concurrency of I&T Schedule			Hi	
Hardware Integration Level		Hi	VHi	VHi
ECONOMIC FACTORS				
Cost Input Base Year			1997	
Purchased Items			0	
AVERAGE MONTHLY LABOR RATE			**15,200**	
Direct Software Management			**15,200**	
Software System Engineering			**15,200**	
Software Design			**15,200**	
Software Programming			**15,200**	
Software Data Preparation			**15,200**	
Software Test			**15,200**	
Software Configuration Management			**15,200**	
Software Quality Assurance			**15,200**	
SOFTWARE MAINTENANCE				
Years of Maintenance			**20**	
Separate Sites			1	
Maintenance Growth Over Life			**10.00%**	
Personnel Differences		Low	Nom-	Nom
Development Environment Difference		Nom	Nom	Nom+
Annual Change Rate			**10.00%**	
Maintenance Level (Rigor)		Nom	Nom	Nom
Min Maintenance Staff (Optional)			0.0	
Max Maintenance Staff (Optional)			0.0	
Maintenance Monthly Labor Rate			**15,200**	
Additional Annual Maintenance Cost			0	
Maintenance Start Date			8/20/98	
Percent To Be Maintained			100.00%	
Maintain Total System			YES	
SOFTWARE CODE METRICS (Optional)				
ESTIMATE TO COMPLETE				
ADJUSTMENT FACTORS				

Appendix E: SoftCost-OO Report

SOFTCOST-OO Project Summary Report Page 1

PROJECT INFORMATION

Project name : SOFTCOST
Estimate date and time: 06/24/98 09:47 am
Version:
Start date: 06/24/98
Number of Subprojects: 4

Calibration File Name: D:\AFIT\THESIS~1\SOFTCOST\THSCADA.CAL
WBS File Name: STANDARD.WBS
Work Holidays File Name: STANDARD.HOL

CALIBRATION COEFFICIENTS

Productivity multiplier (A)	:	1.264
Schedule multiplier (B)	:	3.200
Effort exponent (alpha)	:	1.100
Schedule exponent (beta)	:	0.370
Base effort constant (gamma)	:	3.000
Work hours/person-month	:	152.0

SIZING SUB-MODEL WEIGHTING FACTORS

New OO Components	:	1.000
Re-used OO Components	:	0.200
Modified OO Components	:	0.300
New Other Components	:	1.000
Re-used Other Components	:	0.250
Modified Other Components	:	0.400

BASE ESTIMATE

	Effective Size (KSLOC/FP)	Duration (months)	Effort (pm)	Productivity (SLOC/pm)	Average Staff (persons)	Confidence (%)
SOFTCOST	175.0	44.7	1242.2	144.4	27.8	51.5
CSC 1	20.0	14.1	107.4	161.8	7.6	38.0
CSC 2	30.0	20.0	167.8	172.0	8.4	46.5
CSCI 2	80.0	29.8	493.6	158.5	16.6	47.8
CSCI 3	45.0	23.5	62.1	166.0	11.2	46.9

SOFTCOST PROJECT FACTORS

Type of Software	Avionics (Military)	
System Architecture	Centralized	(1.000)
Number of Organizations Involved	1	
Organizational Interface Complexity	Nominal	
Staff Resource Availability	Nominal	(1.000)
Computer Resource Availability	Nominal	(1.000)
Security Requirements	Nominal	(1.000)

SOFTCOST PROCESS FACTORS

Degree of Standardization	High	(1.190)
Scope of Support	Nominal	(1.000)
Use of Modern Software Methods	Nominal	
Use of Peer Reviews	Nominal	(1.000)
Use of Software Tools/Environment	Nominal	
Software Tool/Environment Stability	Nominal	

SOFTCOST PRODUCT FACTORS

Technology Usage Factor	Nominal	(1.000)
Product Complexity	Nominal	(1.000)
Requirements Volatility	Nominal	(1.000)
Degree of Optimization	Nominal	(1.000)
Degree of Real-Time	Nominal	(1.000)
Re-use Benefits	Nominal	

Re-use Costs	Nominal	
Database Size	Nominal	(1.000)

SOFTCOST PERSONNEL FACTORS

Number of OO Projects Completed	1	
Analyst Capability	Nominal	(1.000)
Applications Experience	Nominal	(1.000)
Environment Experience	Nominal	(1.000)
Language Experience	Nominal	(1.000)
Methodology Experience	Nominal	(1.000)
Team Capability	Nominal	(1.000)

CSC 1 PROJECT FACTORS Inherited

Type of Software	Avionics (Military)	
System Architecture	Centralized	(1.000)
Number of Organizations Involved	1	
Organizational Interface Complexity	Nominal	
Staff Resource Availability	Nominal	(1.000)
Computer Resource Availability	Nominal	(1.000)
Security Requirements	Nominal	(1.000)

CSC 1 PROCESS FACTORS Inherited

Degree of Standardization	High	(1.190)
Scope of Support	Nominal	(1.000)
Use of Modern Software Methods	Nominal	
Use of Peer Reviews	Nominal	(1.000)
Use of Software Tools/Environment	Nominal	
Software Tool/Environment Stability	Nominal	

CSC 1 PRODUCT FACTORS Inherited

Technology Usage Factor	Nominal	(1.000)
Product Complexity	Nominal	(1.000)
Requirements Volatility	Nominal	(1.000)
Degree of Optimization	Nominal	(1.000)
Degree of Real-Time	Nominal	(1.000)
Re-use Benefits	Nominal	
Re-use Costs	Nominal	
Database Size	Nominal	(1.000)

CSC 1	PERSONNEL FACTORS	Inherited
Number of OO Projects Completed	1	
Analyst Capability	Nominal	(1.000)
Applications Experience	Nominal	(1.000)
Environment Experience	Nominal	(1.000)
Language Experience	Nominal	(1.000)
Methodology Experience	Nominal	(1.000)
Team Capability	Nominal	(1.000)

KILO-SOURCE LINES OF CODE (KSLOC's)		MOST		
CSC 1	MAX	LIKELY	MIN	WEIGHTED
New OO Components	20.0	20.0	20.0	20.0
Re-used OO Components	0.0	0.0	0.0	0.0
Modified OO Components	0.0	0.0	0.0	0.0
New Other Components	0.0	0.0	0.0	0.0
Re-used Other Components	0.0	0.0	0.0	0.0
Modified Other Components	0.0	0.0	0.0	0.0

Total effective size : 20.0 KSLOC Size variance: 0.0 KSLOC

CSC 2	PROJECT FACTORS	Inherited
Type of Software	Avionics (Military)	
System Architecture	Centralized	(1.000)
Number of Organizations Involved	1	
Organizational Interface Complexity	Nominal	
Staff Resource Availability	Nominal	(1.000)
Computer Resource Availability	Nominal	(1.000)
Security Requirements	Nominal	(1.000)

CSC 2	PROCESS FACTORS	Inherited
Degree of Standardization	High	(1.190)
Scope of Support	Nominal	(1.000)
Use of Modern Software Methods	Nominal	
Use of Peer Reviews	Nominal	(1.000)
Use of Software Tools/Environment	Nominal	
Software Tool/Environment Stability	Nominal	

CSC 2	PRODUCT FACTORS	Inherited

Technology Usage Factor	Nominal	(1.000)
Product Complexity	Nominal	(1.000)
Requirements Volatility	Nominal	(1.000)
Degree of Optimization	Nominal	(1.000)
Degree of Real-Time	Nominal	(1.000)
Re-use Benefits	Nominal	
Re-use Costs	Nominal	
Database Size	Nominal	(1.000)

CSC 2 PERSONNEL FACTORS Inherited

Number of OO Projects Completed	1	
Analyst Capability	Nominal	(1.000)
Applications Experience	Nominal	(1.000)
Environment Experience	Nominal	(1.000)
Language Experience	Nominal	(1.000)
Methodology Experience	Nominal	(1.000)
Team Capability	Nominal	(1.000)

KILO-SOURCE LINES OF CODE (KSLOC's)

CSC 2	MAX	MOST LIKELY	MIN	WEIGHTED
New OO Components	30.0	30.0	30.0	30.0
Re-used OO Components	0.0	0.0	0.0	0.0
Modified OO Components	0.0	0.0	0.0	0.0
New Other Components	0.0	0.0	0.0	0.0
Re-used Other Components	0.0	0.0	0.0	0.0
Modified Other Components	0.0	0.0	0.0	0.0

Total effective size : 30.0 KSLOC Size variance: 0.0 KSLOC

SOFTCOST-OO Project Summary Report Page 6

CSCI 2 PROJECT FACTORS Inherited

Type of Software	Avionics (Military)	
System Architecture	Centralized	(1.000)
Number of Organizations Involved	1	
Organizational Interface Complexity	Nominal	
Staff Resource Availability	Nominal	(1.000)
Computer Resource Availability	Nominal	(1.000)
Security Requirements	Nominal	(1.000)

CSCI 2 PROCESS FACTORS Inherited

Degree of Standardization	High	(1.190)
Scope of Support	Nominal	(1.000)
Use of Modern Software Methods	Nominal	
Use of Peer Reviews	Nominal	(1.000)
Use of Software Tools/Environment	Nominal	
Software Tool/Environment Stability	Nominal	

CSCI 2 PRODUCT FACTORS Inherited

Technology Usage Factor	Nominal	(1.000)
Product Complexity	Nominal	(1.000)
Requirements Volatility	Nominal	(1.000)
Degree of Optimization	Nominal	(1.000)
Degree of Real-Time	Nominal	(1.000)
Re-use Benefits	Nominal	
Re-use Costs	Nominal	
Database Size	Nominal	(1.000)

CSCI 2 PERSONNEL FACTORS Inherited

Number of OO Projects Completed	1	
Analyst Capability	Nominal	(1.000)
Applications Experience	Nominal	(1.000)
Environment Experience	Nominal	(1.000)
Language Experience	Nominal	(1.000)
Methodology Experience	Nominal	(1.000)
Team Capability	Nominal	(1.000)

KILO-SOURCE LINES OF CODE (KSLOC's)

CSCI 2	MAX	MOST LIKELY	MIN	WEIGHTED
New OO Components	80.0	80.0	80.0	80.0
Re-used OO Components	0.0	0.0	0.0	0.0
Modified OO Components	0.0	0.0	0.0	0.0
New Other Components	0.0	0.0	0.0	0.0
Re-used Other Components	0.0	0.0	0.0	0.0
Modified Other Components	0.0	0.0	0.0	0.0

Total effective size : 80.0 KSLOC Size variance: 0.0 KSLOC

CSCI 3 PROJECT FACTORS		Inherited
Type of Software	Avionics (Military)	
System Architecture	Centralized	(1.000)
Number of Organizations Involved	1	
Organizational Interface Complexity	Nominal	
Staff Resource Availability	Nominal	(1.000)
Computer Resource Availability	Nominal	(1.000)
Security Requirements	Nominal	(1.000)

CSCI 3 PROCESS FACTORS		Inherited
Degree of Standardization	High	(1.190)
Scope of Support	Nominal	(1.000)
Use of Modern Software Methods	Nominal	
Use of Peer Reviews	Nominal	(1.000)
Use of Software Tools/Environment	Nominal	
Software Tool/Environment Stability	Nominal	

CSCI 3 PRODUCT FACTORS		Inherited
Technology Usage Factor	Nominal	(1.000)
Product Complexity	Nominal	(1.000)
Requirements Volatility	Nominal	(1.000)
Degree of Optimization	Nominal	(1.000)
Degree of Real-Time	Nominal	(1.000)
Re-use Benefits	Nominal	
Re-use Costs	Nominal	
Database Size	Nominal	(1.000)

CSCI 3 PERSONNEL FACTORS		Inherited
Number of OO Projects Completed	1	
Analyst Capability	Nominal	(1.000)
Applications Experience	Nominal	(1.000)
Environment Experience	Nominal	(1.000)
Language Experience	Nominal	(1.000)
Methodology Experience	Nominal	(1.000)
Team Capability	Nominal	(1.000)

KILO-SOURCE LINES OF CODE (KSLOC's) CSCI 3	MAX	MOST LIKELY	MIN	WEIGHTED
New OO Components	45.0	45.0	45.0	45.0
Re-used OO Components	0.0	0.0	0.0	0.0

Modified OO Components	0.0	0.0	0.0	0.0
New Other Components	0.0	0.0	0.0	0.0
Re-used Other Components	0.0	0.0	0.0	0.0
Modified Other Components	0.0	0.0	0.0	0.0

Total effective size : 45.0 KSLOC Size variance: 0.0 KSLOC

Appendix F: SoftEst Report

Project Name: SoftEst

Wrap Rate: 100.00
Hours per Man-month: 152

Total Effort: 2703.42
Longest Schedule: 50.64
Total Size: 175000.00
Size Std Dev: 0.00
Total Cost: 41091952.00

Maintenance Settings
Base Year Effort: 270.34
Total Maint. Effort: 5663.66
Calculation Method: 0
Annual Change Traffic: 10
Years to Maintain: 20
ADSI: 175000.00
Assessment & Assimilation: 4
S/W Understanding: 30
Redesign: 15
Recode: 15
Retest: 15

Number of CSCIs: 3

CSCI name: CSCI_1
Model Mode: AdaEMBEDDED
EAF: 2.07
Size: 50000.00
Standard Deviation: 0.00
Effort: 745.79
Cost: 11335989.00
Longest Schedule: 52.28
KDSI: 50.00

Constraint Settings
Constraint Mode: None
User-input Total Schedule: 0.0

Phase	Schedule	FSP
0	9.9	6.7
1	12.9	9.9
2	8.3	19.5
3	5.0	24.7
4	6.9	20.8
5	9.3	13.2

Environmental Parameter Settings

1	Analyst Capability,	NM,	1.0000
2	Programmer Capability,	NM,	1.0000
3	Application Experience,	NM,	1.0000
4	Virtual Machine Experience,	NM,	1.0000
5	Language Experience,	NM,	1.0000
6	Processing Time Constraints,	NM,	1.0000
7	Hardware Storage Constraints,	NM,	1.0000
8	Virtual Machine Volatility,	NM,	1.0000
9	Development Turn Around Time,	NM,	1.0000
10	Requirements Volatility,	NM,	1.0000
11	Required Reliability,	HI,	1.1500
12	Data Base Size,	NM,	1.0000
13	Product Complexity,	NM,	1.0000
14	Design For Reuse,	NM,	1.0000
15	Modern Development Practices,	NM,	1.0000
16	Use Of Automated Tools,	NM,	1.0000
17	Classified Environment,	VL,	1.0000
18	Schedule Constraints,	NM,	1.0000
19	Platform Risk,	VH,	1.8000

CSCI Calculation Results

phase 0, effort 66.8, schedule 9.9,	FSP 6.7,	cost 1015163
phase 1, effort 128.0, schedule 12.9,	FSP 9.9,	cost 1945730
phase 2, effort 161.4, schedule 8.3,	FSP 19.5,	cost 2453311
phase 3, effort 122.4, schedule 5.0,	FSP 24.7,	cost 1861133
phase 4, effort 144.7, schedule 6.9,	FSP 20.8,	cost 2199520
phase 5, effort 122.4, schedule 9.3,	FSP 13.2,	cost 1861133

Number of CSCs: 2

CSC name: CSC_1
Size Type: Source Lines of Code
Coding Language: unspecified
FP-SLOC Conversion: 1
Size: 20000.00
Standard Deviation: 0.00
Most Likely Estimate: 20000.00
Low Estimate: 20000.00
High Estimate: 20000.00
EDSI: 0.00
ADSI: 0.00
DM: 0.00
CM: 0.00
IM: 0.00
Cost: 0.00
Schedule: 0.00
Effort: 0.00

CSC name: CSC_2
Size Type: Source Lines of Code
Coding Language: unspecified
FP-SLOC Conversion: 1
Size: 30000.00
Standard Deviation: 0.00
Most Likely Estimate: 30000.00
Low Estimate: 30000.00
High Estimate: 30000.00
EDSI: 0.00
ADSI: 0.00
DM: 0.00
CM: 0.00
IM: 0.00
Cost: 0.00
Schedule: 0.00
Effort: 0.00
CSCI name: CSCI_2
Model Mode: (null)
EAF: 1.69
Size: 80000.00
Standard Deviation: 0.00
Effort: 1282.16

Cost: 19488872.00
Longest Schedule: 62.17
KDSI: 80.00

Constraint Settings
Constraint Mode: None
User-input Total Schedule: 0.0

Phase	Schedule	FSP
0	11.8	9.7
1	15.3	14.3
2	9.8	28.2
3	5.9	35.7
4	8.3	30.1
5	11.0	19.1

Environmental Parameter Settings

1 Analyst Capability,	NM,	1.0000
2 Programmer Capability,	HI,	0.8600
3 Application Experience,	NM,	1.0000
4 Virtual Machine Experience,	NM,	1.0000
5 Language Experience,	HI,	0.9500
6 Processing Time Constraints,	NM,	1.0000
7 Hardware Storage Constraints,	NM,	1.0000
8 Virtual Machine Volatility,	NM,	1.0000
9 Development Turn Around Time,	NM,	1.0000
10 Requirements Volatility,	NM,	1.0000
11 Required Reliability,	HI,	1.1500
12 Data Base Size,	NM,	1.0000
13 Product Complexity,	NM,	1.0000
14 Design For Reuse,	NM,	1.0000
15 Modern Development Practices,	NM,	1.0000
16 Use Of Automated Tools,	NM,	1.0000
17 Classified Environment,	NM,	1.0000
18 Schedule Constraints,	NM,	1.0000
19 Platform Risk,	VH,	1.8000

CSCI Calculation Results

phase 0, effort 114.8, schedule 11.8,	FSP 9.7,	cost 1745272
phase 1, effort 220.1, schedule 15.3,	FSP 14.3,	cost 3345105

phase 2, effort 277.5, schedule 9.8, FSP 28.2, cost 4217740
phase 3, effort 210.5, schedule 5.9, FSP 35.7, cost 3199666
phase 4, effort 248.8, schedule 8.3, FSP 30.1, cost 3781423
phase 5, effort 210.5, schedule 11.0, FSP 19.1, cost 3199666

Number of CSCs: 1

CSC name: DefaultCSC
Size Type: Source Lines of Code
Coding Language: unspecified
FP-SLOC Conversion: 1
Size: 80000.00
Standard Deviation: 0.00
Most Likely Estimate: 80000.00
Low Estimate: 80000.00
High Estimate: 80000.00
EDSI: 0.00
ADSI: 0.00
DM: 0.00
CM: 0.00
IM: 0.00
Cost: 0.00
Schedule: 0.00
Effort: 0.00

CSCI name: CSCI_3
Model Mode: AdaEMBEDDED
EAF: 2.07
Size: 45000.00
Standard Deviation: 0.00
Effort: 675.47
Cost: 10267090.00
Longest Schedule: 50.64
KDSI: 45.00

Constraint Settings
Constraint Mode: None
User-input Total Schedule: 0.0

Phase	Schedule	FSP
0	9.6	6.3

1	12.5	9.3
2	8.0	18.2
3	4.8	23.1
4	6.7	19.5
5	9.0	12.4

Environmental Parameter Settings

1	Analyst Capability,	NM,	1.0000
2	Programmer Capability,	NM,	1.0000
3	Application Experience ,	NM,	1.0000
4	Virtual Machine Experience,	NM,	1.0000
5	Language Experience,	NM,	1.0000
6	Processing Time Constraints,	NM,	1.0000
7	Hardware Storage Constraints,	NM,	1.0000
8	Virtual Machine Volatility,	NM,	1.0000
9	Development Turn Around Time,	NM,	1.0000
10	Requirements Volatility,	NM,	1.0000
11	Required Reliability,	HI,	1.1500
12	Data Base Size,	NM,	1.0000
13	Product Complexity,	NM,	1.0000
14	Design For Reuse,	NM,	1.0000
15	Modern Development Practices,	NM,	1.0000
16	Use Of Automated Tools,	NM,	1.0000
17	Classified Environment,	VL,	1.0000
18	Schedule Constraints,	NM,	1.0000
19	Platform Risk,	VH,	1.8000

CSCI Calculation Results

phase 0, effort	60.5, schedule	9.6,FSP 6.3,	cost 919441
phase 1, effort	115.9, schedule	12.5,FSP 9.3,	cost 1762262
phase 2, effort	146.2, schedule	8.0,FSP 18.2,	cost 2221982
phase 3, effort	110.9, schedule	4.8,FSP 23.1,	cost 1685642
phase 4, effort	131.1, schedule	6.7,FSP 19.5,	cost 1992122
phase 5, effort	110.9, schedule	9.0,FSP 12.4,	cost 1685642

Number of CSCs: 1
CSC name: DefaultCSC
Size Type: Source Lines of Code
Coding Language: unspecified
FP-SLOC Conversion: 1

Size: 45000.00
Standard Deviation: 0.00
Most Likely Estimate: 45000.00
Low Estimate: 45000.00
High Estimate: 45000.00

EDSI:	0.00
ADSI:	0.00
DM:	0.00
CM:	0.00

IM:	0.00
Cost:	0.00
Schedule:	0.00
Effort:	0.00

Last Calibration Mode Used was AdaEMBEDDED

Calibration Set Used: Default Calibration Set

Development Mode: EMBEDDED
Effort Coefficient: 3.3120
Effort Exponent: 1.20
Schedule Coefficient: 4.3760
Schedule Exponent: 0.32

Development Mode: SEMIDETACHED
Effort Coefficient: 3.9700
Effort Exponent: 1.12
Schedule Coefficient: 3.8000
Schedule Exponent: 0.35

Development Mode: ORGANIC
Effort Coefficient: 3.4644
Effort Exponent: 1.05
Schedule Coefficient: 3.6500
Schedule Exponent: 0.38

Development Mode: AdaEMBEDDED
Effort Coefficient: 6.8000
Effort Exponent: 0.94
Schedule Coefficient: 4.3760

Schedule Exponent: 0.32

Development Mode: AdaSEMIDETACHED
Effort Coefficient: 6.8000
Effort Exponent: 0.94
Schedule Coefficient: 4.3760
Schedule Exponent: 0.32

Development Mode: AdaORGANIC
Effort Coefficient: 6.8000
Effort Exponent: 0.94
Schedule Coefficient: 4.3760
Schedule Exponent: 0.32

Development Mode: OBJECT
Effort Coefficient: 6.8000
Effort Exponent: 0.94
Schedule Coefficient: 4.3760
Schedule Exponent: 0.32

Distribution Set Used: Default Distribution Set

Phase	Effort	Schedule
1	0.12	0.30
2	0.23	0.39
3	0.29	0.25
4	0.22	0.15
5	0.26	0.21
6	0.22	0.28

Appendix G: SPR Knowledge Plan 2.0

Project Task Category Report for KnowledgePLAN 2.0 Page 1

Task Category Overview

Security:NONE
Organization:
Location:
Project:THESIS NEW
Description:
Version:01

An asterisk (*) below indicates estimation of that column is enabled for the task category.

Task	Task Category Code	Plan Dates Start	Plan Dates Finish	Plan FTE	Work Plan	Over-time
System deployment	SystemDeploy	6/25/98	7/8/98	3.00 *	29.95 mo *	0.00 d
[Year 01]						
Service and support [Year 01]						
Field service [Year 01]	FieldService	7/8/98	7/13/99	14.32 *	3,781.31 mo *	0.00 d
Customer support [Year 01]	CustSupport	7/8/98	7/13/99	1.24 *	326.62 mo *	0.00 d
Central maintenance [Year 01]						
Maintenance defect rework preparation [Year 01]	MaintDefRepPrep	7/8/98	7/13/99	0.63 *	166.51 mo *	0.00 d
Maintenance defect rework execution [Year 01]	MaintDefRepExec	7/8/98	7/13/99	1.21 *	319.15 mo *	0.00 d
Maintenance defect rework repair [Year 01]	MaintDefRepRepair	7/8/98	7/13/99	3.56 *	938.95 mo *	0.00 d
Maintenance management [Year 01]	MaintMgmt	7/8/98	7/13/99	4.63 *	1,222.87 mo *	0.00 d
Year 1 Total						
[Year 02]						
Service and support [Year 02]						
Field service [Year 02]	FieldService	7/8/99	7/12/00	12.67 *	3,343.91 mo *	0.00 d
Customer support [Year 02]	CustSupport	7/8/99	7/12/00	1.09 *	288.84 mo *	0.00 d
Central maintenance [Year 02]						
Maintenance defect rework preparation [Year 02]	MaintDefRepPrep	7/8/99	7/12/00	0.56 *	147.30 mo *	0.00 d
Maintenance defect rework execution [Year 02]	MaintDefRepExec	7/8/99	7/12/00	1.07 *	282.32 mo *	0.00 d
Maintenance defect rework repair [Year 02]	MaintDefRepRepair	7/8/99	7/12/00	3.14 *	829.52 mo *	0.00 d
Maintenance management [Year 02]	MaintMgmt	7/8/99	7/12/00	4.05 *	1,070.01 mo *	0.00 d
Year 2 Total						
[Year 03]						
Service and support [Year 03]						
Field service [Year 03]	FieldService	7/7/00	7/12/01	11.01 *	2,906.50 mo *	0.00 d
Customer support [Year 03]	CustSupport	7/7/00	7/12/01	0.95 *	251.06 mo *	0.00 d
Central maintenance [Year 03]						
Maintenance defect rework preparation [Year 03]	MaintDefRepPrep	7/7/00	7/12/01	0.49 *	128.09 mo *	0.00 d
Maintenance defect rework execution [Year 03]	MaintDefRepExec	7/7/00	7/12/01	0.93 *	245.50 mo *	0.00 d
Maintenance defect rework repair [Year 03]	MaintDefRepRepair	7/7/00	7/12/01	2.74 *	723.63 mo *	0.00 d
Maintenance management [Year 03]	MaintMgmt	7/7/00	7/12/01	3.62 *	955.37 mo *	0.00 d
Year 3 Total						
[Year 04]						
Service and support [Year 04]						
Field service [Year 04]	FieldService	7/7/01	7/11/02	9.37 *	2,473.18 mo *	0.00 d
Customer support [Year 04]	CustSupport	7/7/01	7/11/02	0.81 *	213.63 mo *	0.00 d
Central maintenance [Year 04]						
Maintenance defect rework preparation [Year 04]	MaintDefRepPrep	7/7/01	7/11/02	0.41 *	108.87 mo *	0.00 d
Maintenance defect rework execution [Year 04]	MaintDefRepExec	7/7/01	7/11/02	0.79 *	208.67 mo *	0.00 d
Maintenance defect rework repair [Year 04]	MaintDefRepRepair	7/7/01	7/11/02	2.33 *	614.20 mo *	0.00 d
Maintenance management [Year 04]	MaintMgmt	7/7/01	7/11/02	3.04 *	802.51 mo *	0.00 d
Year 4 Total						
[Year 05]						
Service and support [Year 05]						
Field service [Year 05]	FieldService	7/7/02	7/10/03	7.71 *	2,035.78 mo *	0.00 d
Customer support [Year 05]	CustSupport	7/7/02	7/10/03	0.67 *	175.85 mo *	0.00 d
Central maintenance [Year 05]						
Maintenance defect rework preparation [Year 05]	MaintDefRepPrep	7/7/02	7/10/03	0.34 *	89.66 mo *	0.00 d
Maintenance defect rework execution [Year 05]	MaintDefRepExec	7/7/02	7/10/03	0.65 *	171.85 mo *	0.00 d
Maintenance defect rework repair [Year 05]	MaintDefRepRepair	7/7/02	7/10/03	1.91 *	504.77 mo *	0.00 d
Maintenance management [Year 05]	MaintMgmt	7/7/02	7/10/03	2.46 *	649.65 mo *	0.00 d
Year 5 Total						
Grand Total				**19.76**	**1,182.09 d**	**0.00mo**

Project Task Category Report for KnowledgePLAN 2.0 Page 2

Task Category Overview

Security:NONE
Project:THESIS NEW
Organization:
Description:
Location:
Version:01

An asterisk (*) below indicates estimation of that column is enabled for the task category.

Task	Task Category Code	Cost Plan	OT	Fixed	Plan Deliverable
System deployment	SystemDeploy	$23,960	$0	$0	3,555.00 fp *
[Year 01]					
Service and support [Year 01]					
Field service [Year 01]	FieldService	$3,025,048	$0	$0	925.00 fp *
Customer support [Year 01]	CustSupport	$261,296	$0	$0	925.00 fp *
Central maintenance [Year 01]					
Maintenance defect rework preparation [Year 01]	MaintDefRepPrep	$133,208	$0	$0	45.50 kl *
Maintenance defect rework execution [Year 01]	MaintDefRepExec	$255,320	$0	$0	45.50 kl *
Maintenance defect rework repair [Year 01]	MaintDefRepRepair	$751,160	$0	$0	266.00 df *
Maintenance management [Year 01]	MaintMgmt	$978,296	$0	$0	32.00 pe *
Year 1 Total		**$5,404,328**			
[Year 02]					
Service and support [Year 02]					
Field service [Year 02]	FieldService	$2,675,128	$0	$0	818.00 fp *
Customer support [Year 02]	CustSupport	$231,072	$0	$0	818.00 fp *
Central maintenance [Year 02]					
Maintenance defect rework preparation [Year 02]	MaintDefRepPrep	$117,840	$0	$0	40.25 kl *
Maintenance defect rework execution [Year 02]	MaintDefRepExec	$225,856	$0	$0	40.25 kl *
Maintenance defect rework repair [Year 02]	MaintDefRepRepair	$663,616	$0	$0	235.00 df *
Maintenance management [Year 02]	MaintMgmt	$856,008	$0	$0	28.00 pe *
Year 2 Total		**$4,769,520**			
[Year 03]					
Service and support [Year 03]					
Field service [Year 03]	FieldService	$2,325,200	$0	$0	711.00 fp *
Customer support [Year 03]	CustSupport	$200,848	$0	$0	711.00 fp *
Central maintenance [Year 03]			$0	$0	
Maintenance defect rework preparation [Year 03]	MaintDefRepPrep	$102,472	$0	$0	35.00 kl *
Maintenance defect rework execution [Year 03]	MaintDefRepExec	$196,400	$0	$0	35.00 kl *
Maintenance defect rework repair [Year 03]	MaintDefRepRepair	$578,904	$0	$0	205.00 df *
Maintenance management [Year 03]	MaintMgmt	$764,296	$0	$0	25.00 pe *
Year 3 Total		**$4,168,120**			
[Year 04]					
Service and support [Year 04]					
Field service [Year 04]	FieldService	$1,978,544	$0	$0	605.00 fp *
Customer support [Year 04]	CustSupport	$170,904	$0	$0	605.00 fp *
Central maintenance [Year 04]					
Maintenance defect rework preparation [Year 04]	MaintDefRepPrep	$87,096	$0	$0	29.75 kl *
Maintenance defect rework execution [Year 04]	MaintDefRepExec	$166,936	$0	$0	29.75 kl *
Maintenance defect rework repair [Year 04]	MaintDefRepRepair	$491,360	$0	$0	174.00 df *
Maintenance management [Year 04]	MaintMgmt	$642,008	$0	$0	21.00 pe *
Year 4 Total		**$3,536,848**			
[Year 05]					
Service and support [Year 05]					
Field service [Year 05]	FieldService	$1,628,624	$0	$0	498.00 fp *
Customer support [Year 05]	CustSupport	$140,680	$0	$0	498.00 fp *
Central maintenance [Year 05]					
Maintenance defect rework preparation [Year 05]	MaintDefRepPrep	$71,728	$0	$0	24.50 kl *
Maintenance defect rework execution [Year 05]	MaintDefRepExec	$137,480	$0	$0	24.50 kl *
Maintenance defect rework repair [Year 05]	MaintDefRepRepair	$403,816	$0	$0	143.00 df *
Maintenance management [Year 05]	MaintMgmt	$519,720	$0	$0	17.00 pe *
Year 5 Total		**$2,902,048**			
Grand Total		**$20,780,864**	**$0**	**$0**	

SPR KnowledgePLAN Maintenance Attribute Inputs

SPR KnowledgePLAN Page 1 of 1
MAINTENANCE ATTRIBUTES
Security: NONE **Project:** SPR Thesis
Organization: **Description:**
Location: **Version:** 01

	Personnel	
1683	Maintenance Personnel STAFFING	3
1684	Maintenance Personnel EXPERIENCE	3
1685	Maintenance Personnel EDUCATION	3
	Personnel Average	**3**
	Technology	
1699	Maintenance platform computing support	3
1700	Release control methods	3
1701	Problem tracking and reporting	3
1702	Replacement and restructure planning	3
	Technology Average	**3**
	Process	
1643	Centeral Maintenance	3
1644	Field Maintenance	3
1645	Software warranty coverage	3
1646	Customer support	3
1647	Delivery Support	3
	Process Average	**3**
	Environment	
1612	Installation and Production Geography	3
1614	Number of system installation sites	3
1615	Annual growth in installation sites (percent)	3
1616	Number of system maintenance sites	3
	Environment Average	**3**
	Product	
1611	Program execution frequency	3
1609	Current system status	3
1610	Long range product stability	3

SPR KnowledgePLAN 2.0 Project Calibration Inputs

PROJECT CALIBRATION

Security: NONE **Project:** THESIS NEW
Organization: **Description:**
Location: **Version:** 01

Resources

Calendar:	**SPR Base Calendar**
Overtime Percent of Plan Work:	**0.00**
Global Resource Adjustment:	**1.00**
Delete Non-Estimated Resource Assignments?	**No**

Tasks

Task Inclusion	
Automatically Add Tasks?	**No**
Automatically Delete Tasks?	**No**
Estimate Excluded Tasks?	**Yes**
Estimate Dependencies?	**Yes**

Calculation

Automatic Estimation	
Enable Automatic Estimation?	**Yes**
Perform Schedule and Summarize also?	**Yes**
Automatic Scheduling	
Enable Automatic Scheduling?	**Yes**

Potential Defects

Re-estimate Potential Defects?	**Yes**

References

Analytic Sciences Corporation, The. The AFSC Cost Estimating Handbook. Reading MA: prepared for USAF, Air Force Systems Command (AFSC), 1986.

Arthur, Jay. Software Evolution -The Software Maintenance Challenge. LifeStar Publishing, Denver CO, 1997.

Belcher, Larry. "Software process improvements help Oklahoma ALC increase productivity, reduce costs," *Leading Edge,* February 1996.

Bischoff, Col Ron. "Design and Planning Make High-Tech F-22 Easy to Maintain and Support," *Aviation Week & Space Technology*, July 15, 1991.

Boehm, Barry W. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall Inc., 1981.

-------, "Software Risk Management," IEEE Computer Society Press, Los Alamitos CA, 1989.

------- and Philip N. Papaccio. "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering, 14*: 1462 - 1477 (October 1988).

Boulware, Gary W., Belinda J. Nethery, and Bryan D. Turner. "Maintenance Software Model Assumptions Versus Budgeting Realities," *Estimator,* Spring 1991. Stewart, Rodney D. and Richard M. Wyskida. Cost Estimator's Reference Manual. New York NY: John Wiley & Sons, Inc., 1987.

Brooks, Frederick P. "The Mythical ManMonth," Addison Wesley, 1975. An updated and expanded edition was published in 1995.

Coggins, George A. and Roy C. Russell. Software Cost Estimating Models: A Comparative Study of What the Models Estimate. MS Thesis, AFIT/GCA/LAS/ 93S-4. School of Systems and Logistics, Air Force Institute of Technology, Wright Patterson AFB OH, September 1993 (ADA-275989).

Coleman, Don, Dan Ash, Bruce Lowther and Paul Oman. "Using Metrics to Evaluate Software System Maintainability," *IEEE Computer, 27*, No. 8, August 1994: 44-49.

Coleman, Don, Bruce Lowther and Paul Oman. "The Application of Software Maintainability Models in Industrial Software Systems." *Journal of Systems Software 29*, No. 1, April 1995: 3-16.

Department of the Air Force. Guidelines for Successful Acquisition and Management of Software Intensive Systems, Volumes 1 and 2. Software Technology Support Center, Hill Air Force Base UT, June 1996.

Department of Defense. Military Standard. Defense System Software Development. DoD STD-2167A. Washington DC: GPO, 29 February 1988.

Department of Defense. Parametric Cost Estimating Handbook- Joint Government/ Industry Initiative, US Navy, Naval Sea Systems Command, Arlington VA, Fall 1995.

Engle, Charles B. "Why Use ADA for DoD Software Procurement," Presentation Slide Script-DCA100-93-D0066, Delivery Order -0045, March, 1996.

Fain, Lt Gen Jim, as quoted by Lt Gen Robert H. Ludwig. "The Role of Technology in Modern Warfare," *Software Technology Conference, 1992, Salt Lake City Utah*, April 14, 1992.

Ferens, Daniel V. "New Perspectives in Software Logistics Support," *Logistics Spectrum*, 4-8 (Spring 1992).

Ferens, Daniel V. "Evaluation of Eight Software Support Cost Models," *Estimator*, Spring 1991.

Ferens, Daniel V. and Robert B. Gurner. "An Evaluation of Three Function Point Models for Estimation of Software Effort," *NAECON Conference, 1992 Dayton, Oh*, May 1992.

Gerlich, Rainer and Ulrich Denskat. "A Cost Estimation Model for Maintenance and High Reuse, "*Proceedings of the European Software Cost Modeling Conference,* 1994, Ivrea, Italy, May 1994.

Glass, Robert L. "Software Reliability Guidebook," Prentice-Hall, Englewood Cliffs NJ, 1979.

Glass, Robert R. and Ronald A. Noiseux. "Software Maintenance Guidebook." Prentice-Hall, Englewood Cliffs NJ, 1981.

Halstead, Maurice H. "Elements of Software Science," Elsevier: New York, 1977.

Jones, Capers. "Backfiring: Converting Lines of Code to Function Points." *IEEE Computer, 28*, No. 11, November 1995:87-88.

KnowledgePLAN 2.0 User's Manual, Software Productivity Research, Burlington MA, 1997.

Maibor, David S. "Software Acquisition for the 90's: One Big Dilemma," *Crosstalk*, July 1997.

Martin, Roger J. and Wilma M. Osborne. Guidance on Software Maintenance, U.S. Department of Commerce, Computer Science and Technology, NBS Special Publication 500-106, December 1983.

Martin, James and Carma McClure. "Software Maintenance: The Problem and Its Solutions," Prentice Hall, Englewood Cliffs NJ, 1983.

Marzo, David B. Calibration and Validation of the SAGE Software Cost/Schedule Estimating System to United States Air Force Databases. MS Thesis, AFIT/GCA/LSG 97S-6. School of Logistics and Acquisition Management, Air Force Institute of Technology, Wright-Patterson AFB OH, September 1997 (ADA-329958).

McGibbon, Thomas. "A Business Case for Software Process Improvement", Data and Analysis Center for Software, September 1996.

Pigoski, Thomas. *Maintenance,* March, 1994.

PRICE-S Reference manual, Price Systems, Moorestown NJ, Martin Marietta, 1993.

SASET. Version 3.0, IBM, disk. Computer software tutorial. Martin Marietta Corporation, Denver CO, 1990.

Schwenke, Robert S. Class handout, COST 291, Introduction to Cost Analysis. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, Summer Short Quarter 1992.

SEER-SEM User's Manual 4.5, GA SEER Technologies, El Segundo CA.

SoftCost-OO MS/DOS, Software Version 3.1, Manual Revision- 1994, Resource Calculations Inc., Englewood CO, July 1994.

Software Maintenance and Reengineering, IEEE Computer Society Press, Los Alamitos CA, March 1997.

Stutzke, Richard D. "Software Estimating Technology: A Survey," *CrossTalk*, May 1996.

Tilley, Scott R. "Perspectives on Legacy System Reengineering," D R A F T | Version 0.3, Reengineering Center Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, 1995.

VanDoren, Edmond. "Maintenance of Operational Systems—An Overview," Software Technology Review. Kaman Sciences, Colorado Springs CO, January 1997.

VanDoren, Edmond. "Maintainability Index Technique for Measuring Program Maintainability," *Software Technology Review*. Kaman Sciences, Colorado Springs CO, January 1997.

Vigder M. R. and A.W. Kark. "Software Cost Estimation and Control," National Research Council of Canada, February 1994

Welker, Kurt D. and Paul W. Oman. "Software Maintainability Metrics Models in Practice." Crosstalk, *Journal of Defense Software Engineering 8*, 11 (November/December 1995): 19-23.

Wellman, Frank. "Software Costing: An Objective Approach to Estimating and Controlling the Cost of Computer Software." New York: Prentice-Hall, Inc., 1992.

Vita 1st Lieutenant Kevin L. Brummert

Lieutenant Kevin L. Brummert was born on 26 October 1970 in Barberton, Ohio. He graduated from Barberton High School in 1989. In May 1995, he graduated from The University of Akron with a Bachelor of Science in Accountancy. He received his Air Force commission through the Reserve Officer Training Corps Program.

Lieutenant Brummert's first Air Force assignment was at the 88th ABW Cost Analysis and Services Branch, Wright-Patterson AFB, Ohio from June 1995 to May 1997. While there, he served as a base level Cost Analyst.

Lieutenant Brummert then entered the Air Force Institute of Technology's School of Logistics and Acquisition Management Graduate Cost Analysis program in May 1997. Upon graduation, he anticipates a follow-on assignment to the Electronic Systems Center at Hanscom AFB, Massachusetts.

Permanent Address: 198 Shenandoah Blvd
Barberton, OH 44203

Vita 1st Lieutenant Philip R. Mischler, Jr.

First Lieutenant Philip Mischler, Jr. was born on 18 May 1966 in Charlotte, North Carolina. He graduated from East Forsyth High School, North Carolina in 1984. He enlisted in the Air Force in September 1984 and spent several years as an F-16 maintenance technician and instructor at Shaw AFB, South Carolina. He received the degree of Bachelor of Science from North Carolina A&T State University and was commissioned from AFROTC Detachment 605 in 1995. After commissioning, he served two years as a Financial Manager at the Aerospace Systems Center at Wright-Patterson AFB, Ohio. He then entered the School of Logistics and Acquisition Management, Air Force Institute of Technology, in May 1997.

Permanent Address: 1908 Cartwright Drive
Kernersville, NC 27284

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1998	Master's Thesis

4. TITLE AND SUBTITLE
SOFTWARE SUPPORT COST ESTIMATING MODELS:
A COMPARATIVE STUDY OF MODEL CONTENT AND PARAMETER SENSITIVITY

5. FUNDING NUMBERS

6. AUTHOR(S)
Kevin L. Brummert, Lieutenant, USAF
Philip Mischler, Jr, Lieutenant, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
2950 P Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCA/LAS/98S-3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
SMC/FMC
2430 East El Segundo Boulevard, Suite 2010
Los Angeles, CA 90245-4687

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 Words*)

This research entailed a comparison of five software estimating models: PRICE-S, SEER-SEM, SoftCost-OO, SoftEst, and SPR KnowledgePLAN. The objective was to research the differences of the software models as related to software support cost. The following major question areas were addressed: (1) How do the differences between the models impact the resulting cost estimates? (2) To what degree can we explain and adjust for the differences between cost models? All items were for flight avionics of a manned aircraft. The differences between the models significantly impact the resulting estimates. Over the five models evaluated, a range of over $60 million occurred during a twenty year estimate. The researchers can explain the differences in the models due to the different algorithms used, but were not able to normalize the models to achieve equivalent estimates. The researchers feel a typical user will not be able to normalize separate models and should, therefore, concentrate on learning one or two models in detail. Different models are more appropriate depending on the task or project being estimated.

14. Subject Terms	15. NUMBER OF PAGES
Cost Analysis, Cost Estimates, Software, Models, Cost Models, Cost Overruns, Support Cost, Maintenance Costs, Monte Carlo Method, Sampling, Software Engineering	172
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AFIT Control Number AFIT/GCA/LAS/98S-3

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. **Please return completed questionnaire** to: AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT-PATTERSON AFB OH 45433-7765. Your response is **important.** Thank you.

1. Did this research contribute to a current research project? a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?
a. Yes b. No

3. **Please estimate** what this research would have cost in terms of manpower and dollars if it had been accomplished under contract or if it had been done in-house.

Man Years_____ $_____

4. Whether or not you were able to establish an equivalent value for this research (in Question 3), what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments (Please feel free to use a separate sheet for more detailed answers and include it with this form):

_____ _____
Name and Grade Organization

_____ _____
Position or Title Address